🇺🇸English Version

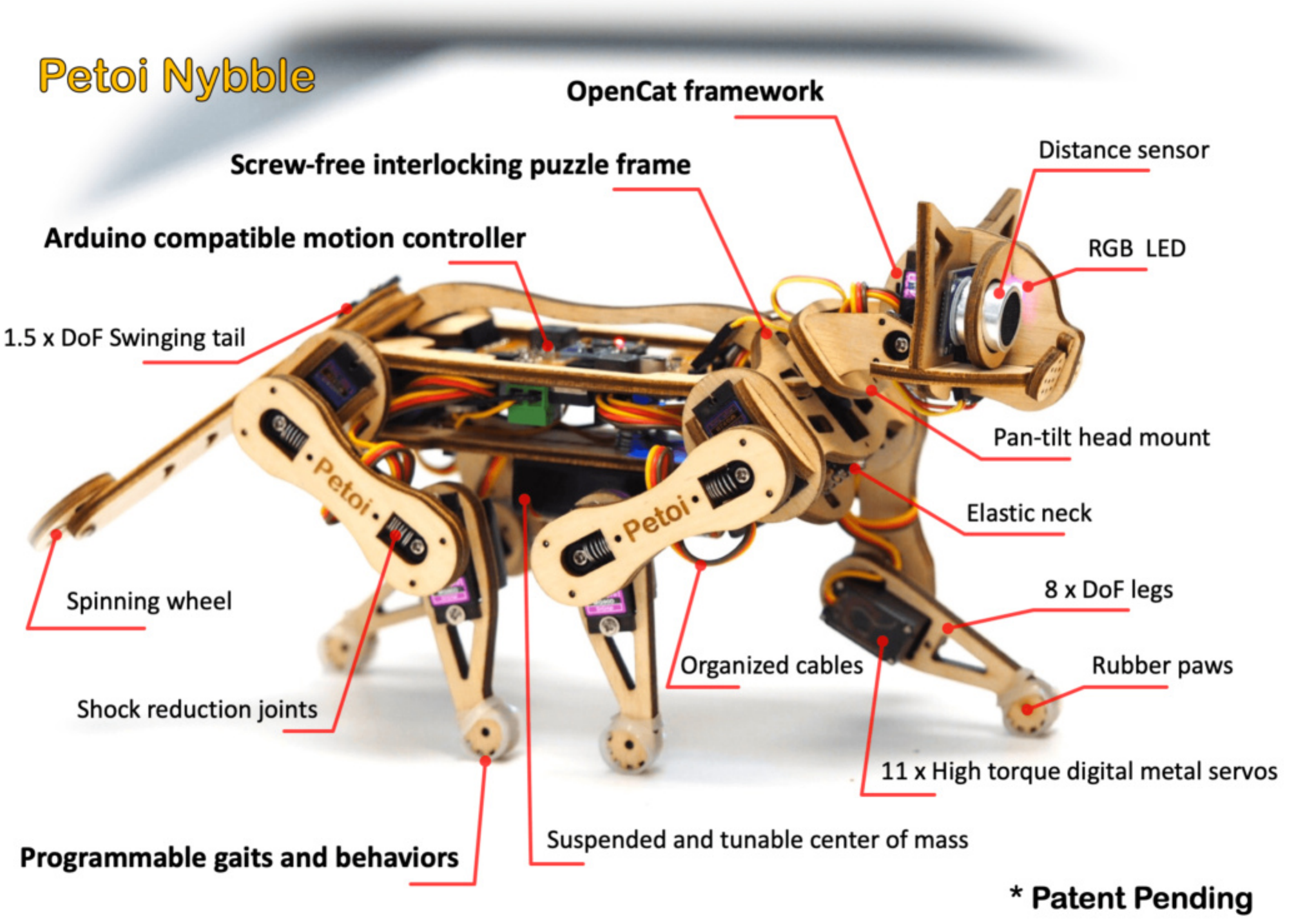
Nybble User Manual 🐱

Rongzhong Li

Getting started

To keep these instructions simple to use, I'm focusing on the assembly rather than an in-depth explanation.

If you have specific questions on "why" rather than "how", please post on our forum at https://www.petoi.com/forum or write to info@petoi.com.



You can support us by shopping at www.petoi.com/store. Our social media (Instagram/Twitter/Facebook/GitHub) account is **@PetoiCamp**. Share your build with us by tagging **#nybble #petoi #opencat** so that we can repost for you!

> ⓘ Last Updated: 01/03/2022

1 🧔‍♂ Tools and Preparation

"A beard well lathered is half shaved."

1.1. Preparation

Prepare a clean desk and some small boxes to unzip the package. Take a picture of the kit contents in case you lost something later.

> It’s better to work in a room without carpet or textured mosaic. Little screws and springs can magically hide themselves if dropped onto the ground.

1.2. Tools and accessories

Required

Tool	Notes
Utility knife	Cut the tabs holding the wooden puzzle pi
Sanding paper/foam	Remove pointy fibers
Flat and Phillips screwdrivers	For M2 (diameter = 2mm) screws
Computer with Arduino IDE	Install the latest Arduino IDE
USB to mini USB cable	Connect the uploader to computer. Not mic
2 x 14500 3.7V Li-ion batteries	Don’t mix with regular AA batteries (1.5V)! The capacity should be around 800mAh.
Smart charger for batteries	Wrong charger may cause danger!

Optional

Tool	Note
Soldering iron w/ accessories	Solder the decorative LED to ultrasound se

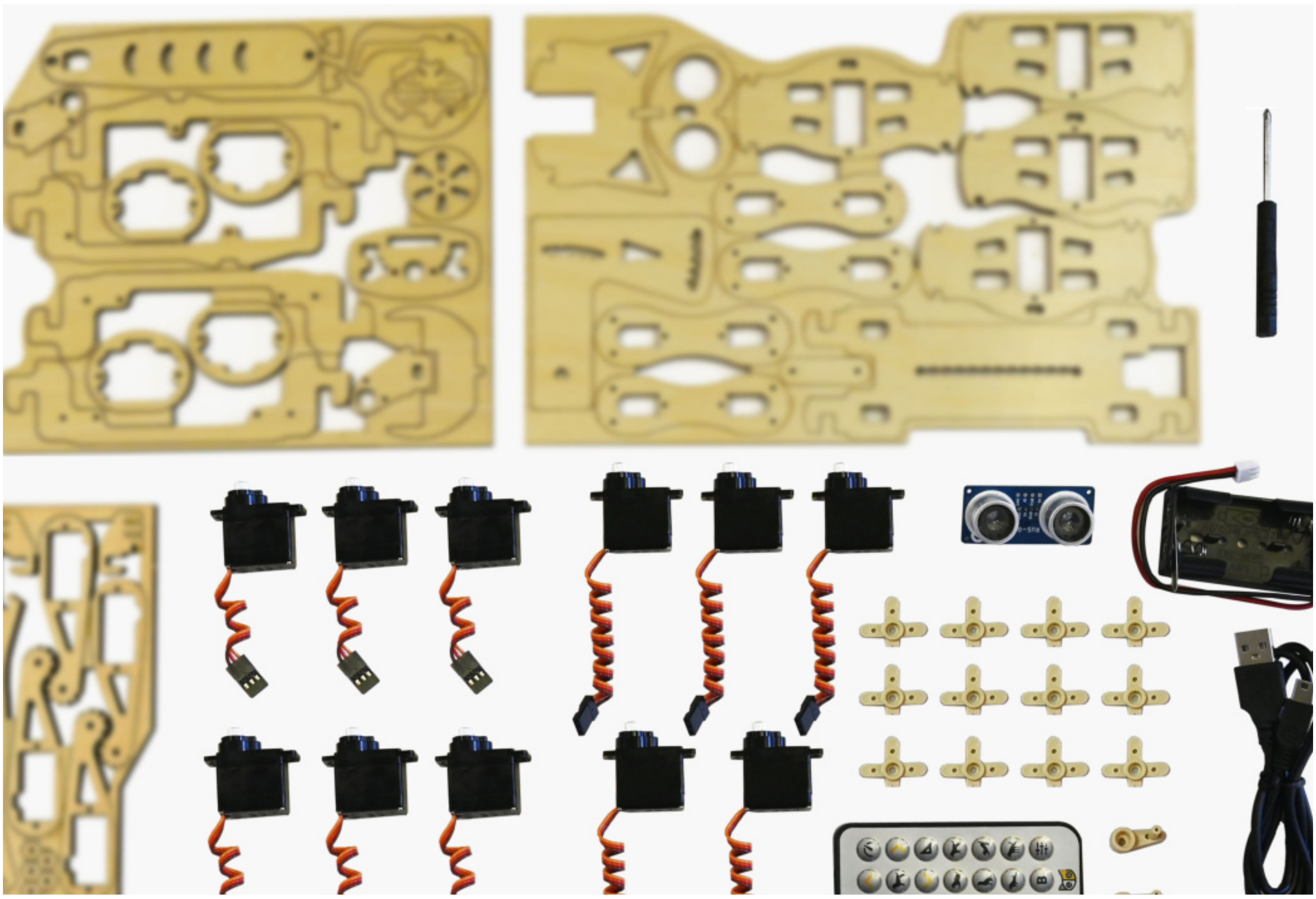
HC-05 bluetooth module	Wirelessly upload sketch and communicat
Color paints	Give your Nybble a unique look
3D printer w/ accessories	Add your special design
Arduino/Raspberry Pi kit	Add more gadgets to Nybble
Multimeter	Test and debug
Oscilloscope	Test and debug
Hot glue/super glue	Avoid using them. OpenCat is designed to

2 Open the Box

"life is like a box of Nybble."

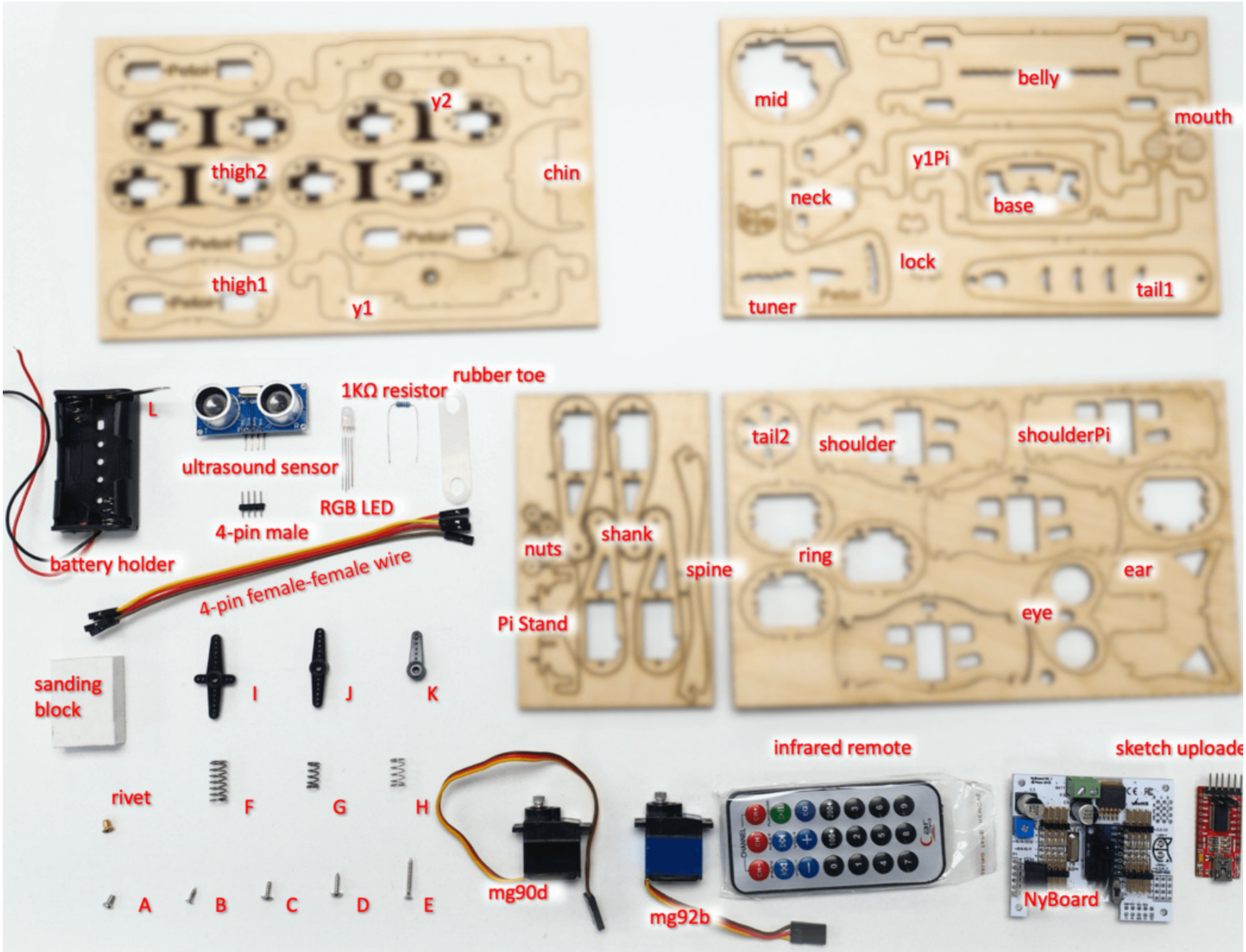
The actual kit contents and packaging method may be adjusted as we improve the product. This instruction will keep consistent with the current namespace.

Newest kit contents:





Early kit contents:



2.1. Cut body pieces off the baseboard.

There might be some tar residue on the wooden pieces from laser cutting. Use a piece of wet soft tissue to clean up the board.

The functional pieces are attached to the baseboard by lightly cut tabs. Though you could pop those pieces out by hand, it's very highly recommended that you use a knife to cut on the backside of the tabs to avoid potential damage to the middle layer, where the fiber direction is perpendicular to the surface fiber.

After taking out all the pieces from the baseboard, you are encouraged to bend and break the remaining structures on the baseboard, to understand the mechanical properties of plywood, such as anisotropic

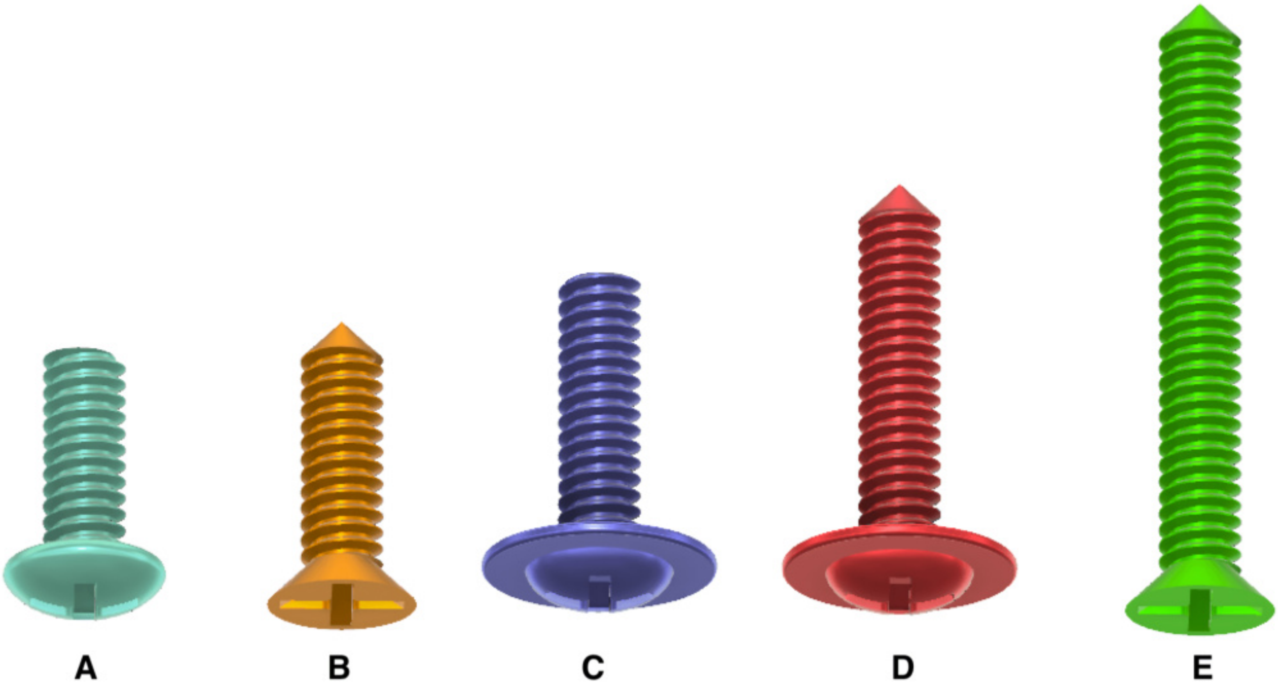
strength, elasticity, etc. That will give you confidence in later handling.

2.2. Remove pointy fibers.

Use the sanding foam to clean up any thorn on the pieces. Don't sand too much or it may affect the tightness between joints.

2.3. Screws

There are five different screws used in the kit. I'm coloring them differently to better indicate their locations. Not all screws are required to assemble Nybble. Not all holes on the puzzle pieces need screws. Observe the assembling animation carefully to locate them.

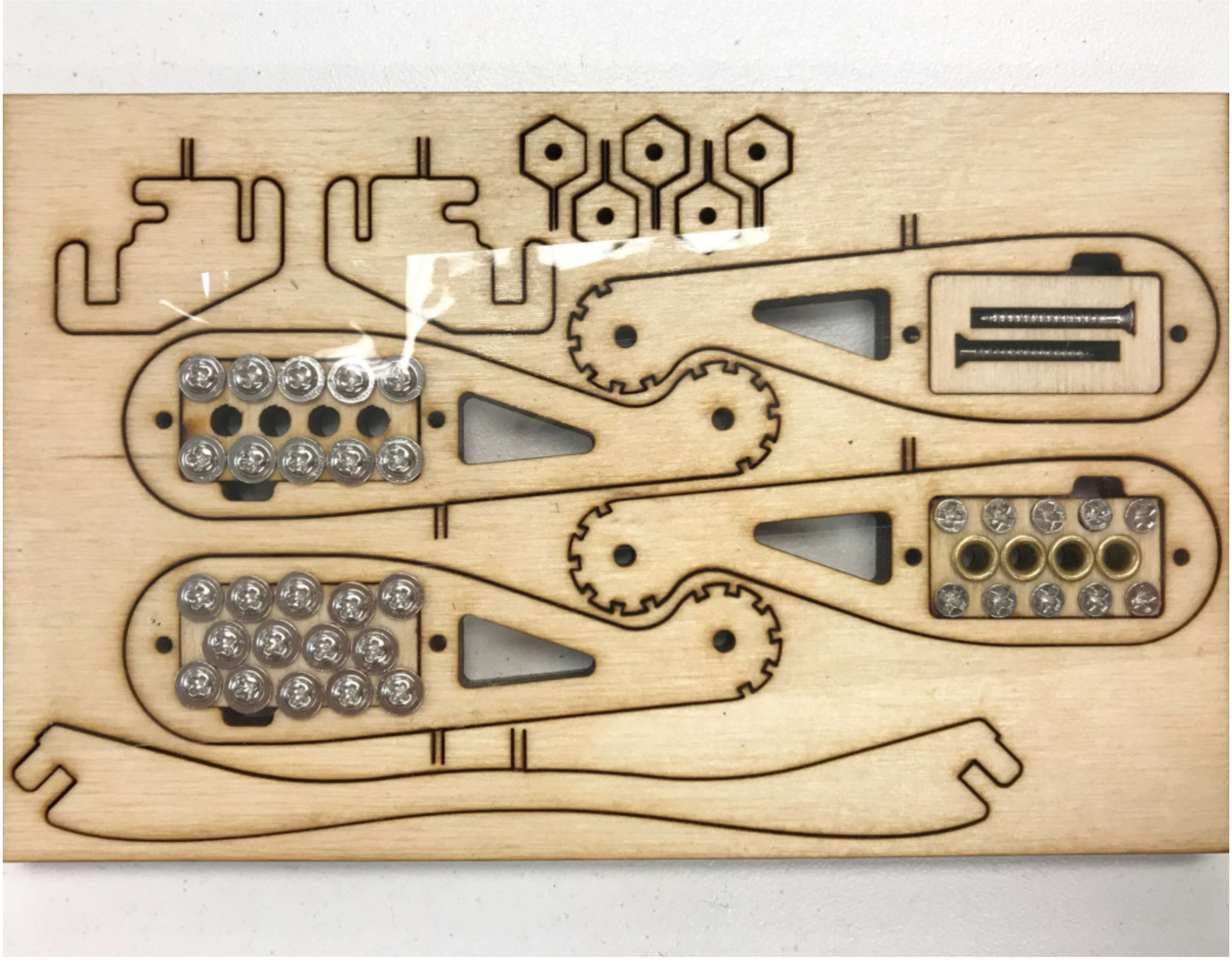


- A is for attaching servo arms. D (sharp tip) is for attaching servos to the frame. A and D come in each servo's accessory pouch with plastic servo arms.
- B is for attaching servo arms/circuit boards to the frame.

> ⓘ In later versions, it may be replaced by C to simplify the kit content. In that case, if the hole is too small for screw C's flat tip, use screw D to pre-tap.

- C (flat tip) is for binding the thighs.
- E (always the longest) is for attaching the battery holder.

For earlier packages, B, C and E are located in the shank board within the multi-punched blocks like this:



2.4. Springs

There are three different springs: F, G, H.

- The big spring F is used for elastic connection in the thigh. There's one spare unit.
- The hard short spring G is for the neck

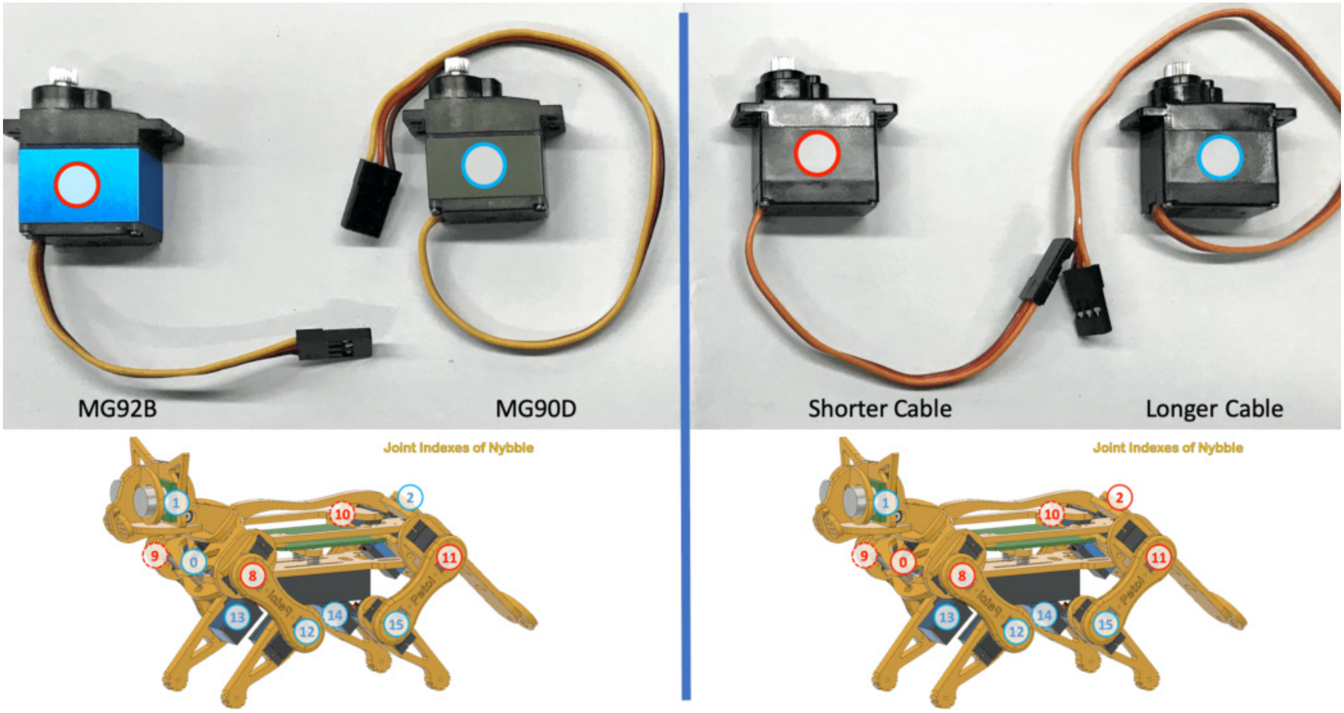
> ⓘ It's replaced by spring F in later versions

- The soft short spring H is for attaching the battery holder.

2.5. Servos

We are switching to a new servo manufacturer from recent batches. Previously, MG92B was used for the four shoulder joints. MG90D was used for other joints.

The new servos are differentiated by their cable length. Shorter cables are used for the neck, tail, and four shoulder joints. Longer cables are used for head tilting and the four knee joints.



> ⓘ For hobbyist servos, there are several fields where they can differentiate.
>
> In the Nybble kit, we are using ODMed metal gear, digital PWM, HV servos with bearing, and brushed iron core motors. Other generic servos can still work with the OpenCat framework but may need more trials and errors for best performance.

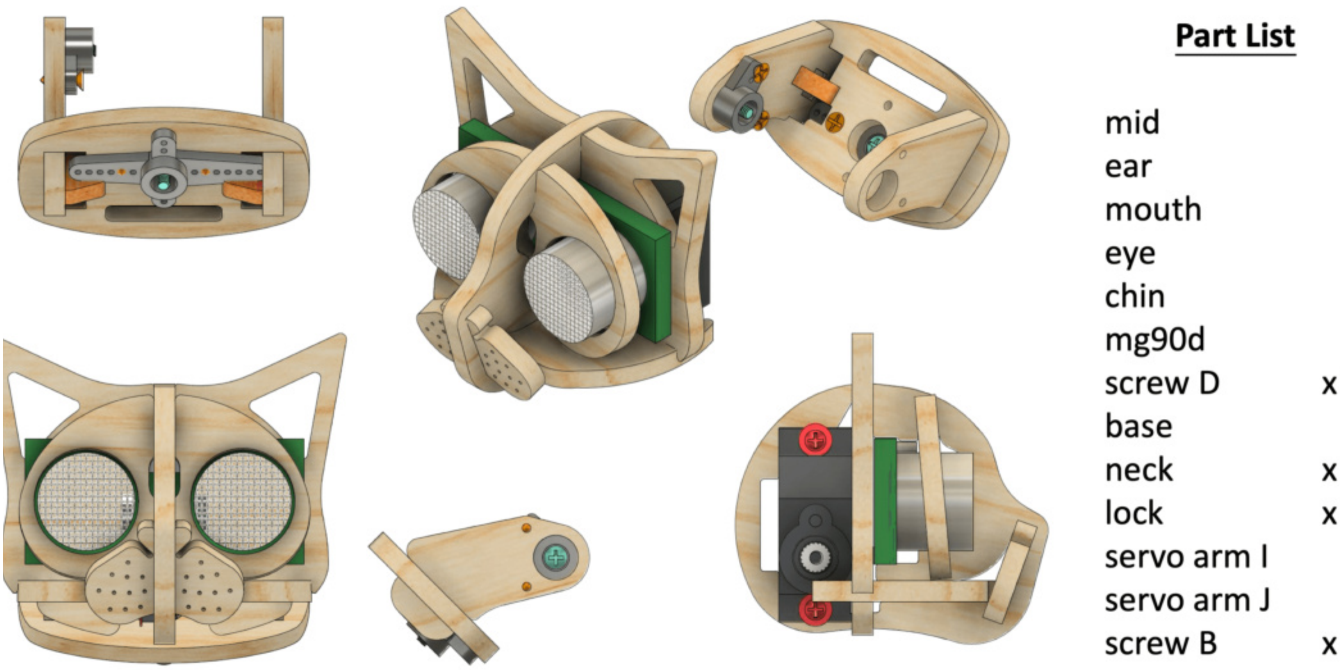
Gear	Signal	Protocol	Voltage	Motor	Bearing
Plastic	Analog	PWM	5V	Brushed	None
Metal	Digital	Serial	HV (up to 8V)	Coreless	Yes
				Brushless	

3 Assemble the Frame

"The whole is more than the sum of its parts."

3.1. Head and neck

3.1.1. Part list



3.1.2. Prepare ultrasonic sensor

> ⓘ The obstacle avoidance algorithm using the ultrasound sensor has not yet been integrated into the released code. The following setup provides a start point but is not mandatory.

> ⓘ After Jan.1st, 2021, we include an ODMed ultrasonic sensor with RGB LED in the Nybble kit. It doesn't require any soldering, but you need to trim off the clip residue of the connection wire on the head's end.
>
> You can download the demo code at https://github.com/PetoiCamp/OpenCat/tree/main/ModuleTests/testRGBLedUlrasonic
>
> The tutorial video is at https://youtu.be/kV2UYbpfGic

The sensor is connected to the NyBoard via a 4 pin cable.

- **Solder on the optional LED to the ultrasonic sensor**

The optional RGB LED can be soldered to the four pins of the ultrasound sensor (instructions) to indicate its working status or can be programmed as decorative light.

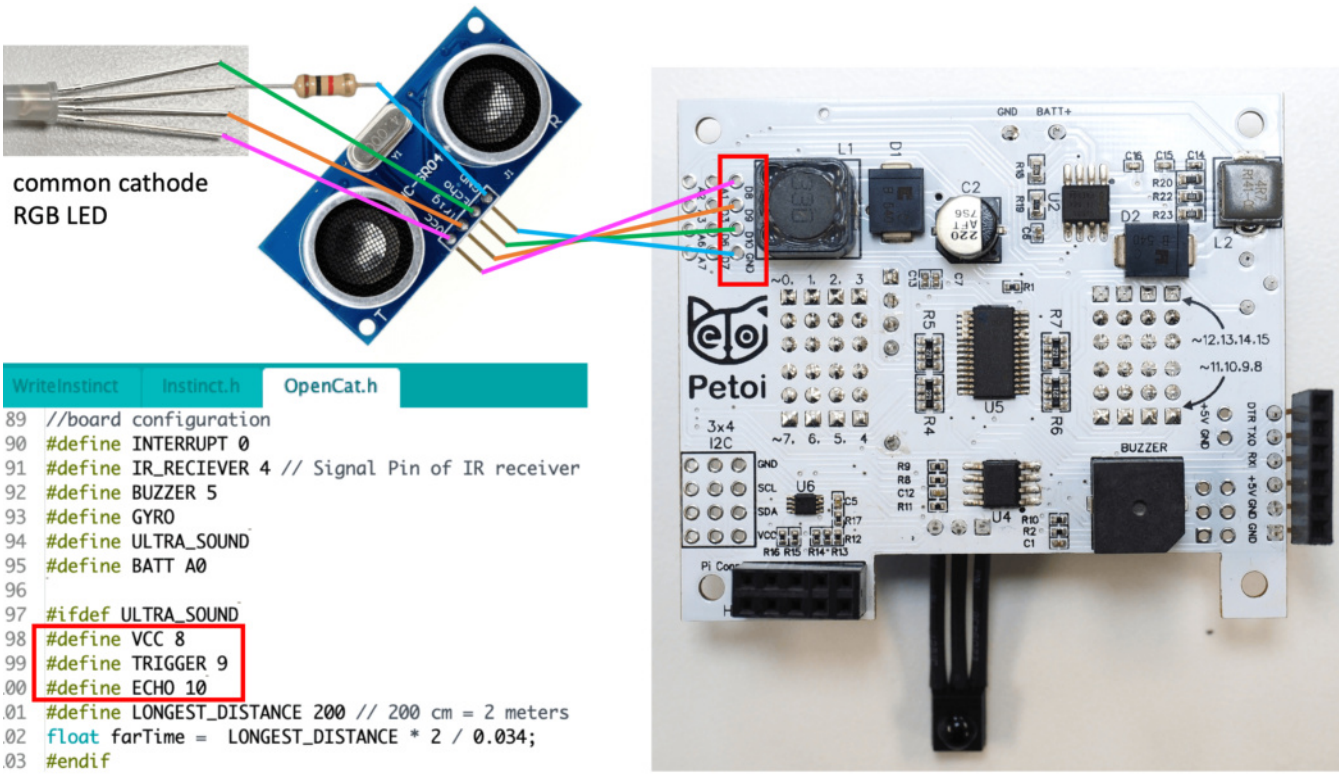
Bend the pins of the ultrasonic sensor for later installation.





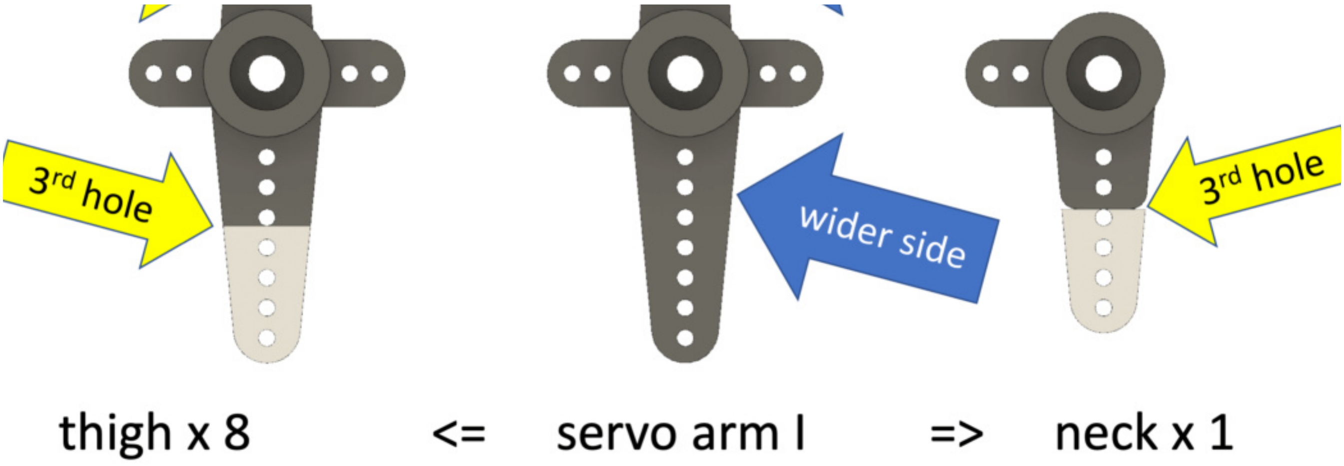
- **Solder 4-pin ultrasonic sensor header to NyBoard**

The ultrasonic module is connected to the exposed GPIO pins located in the opposite corner of the board from the TTL connector. You can customize the pin definitions in **OpenCat.h**. By default definitions, solder the ultrasonic module connection header in the holes labeled “D8 D9 D10 GND” as pictured below.



3.1.3. Trim the servo arms for attaching servos.



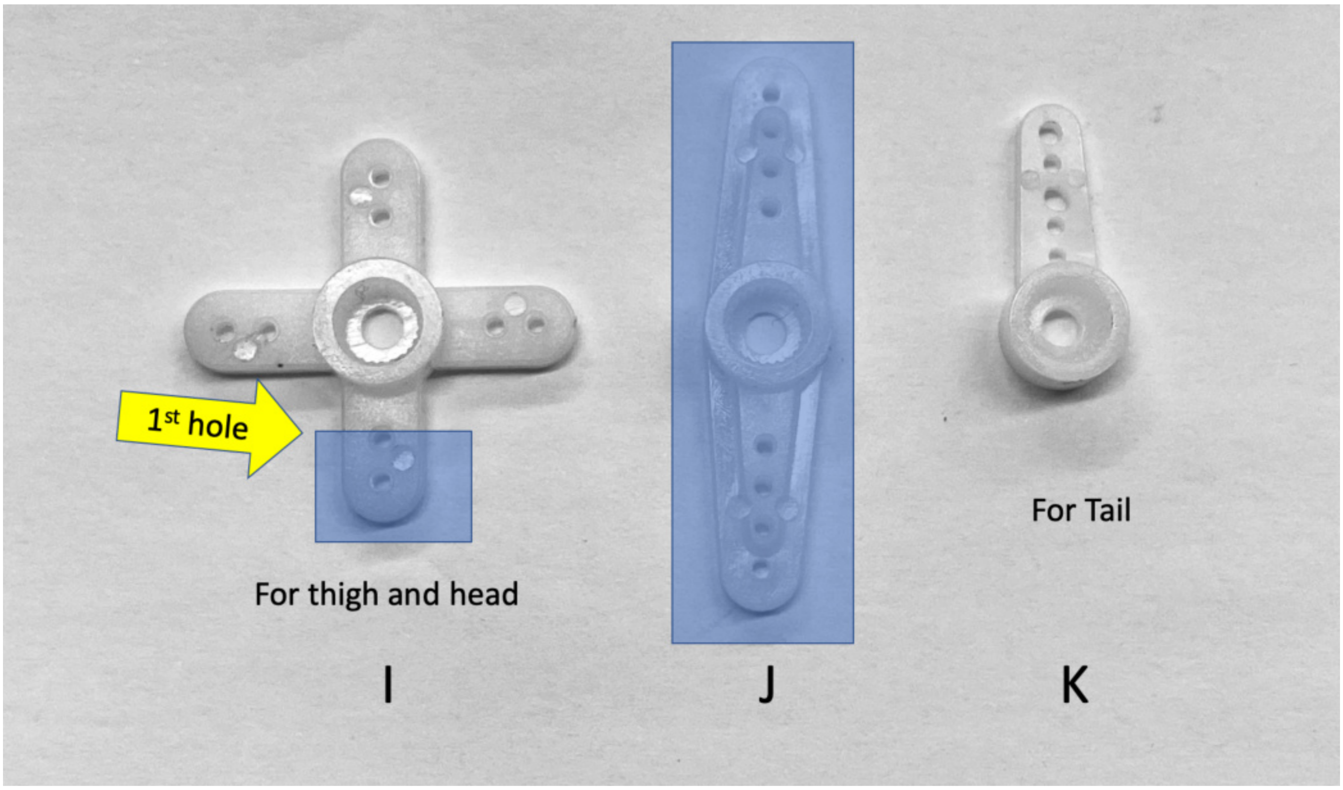


Most of the servo arms on the model are trimmed from the cross-shaped arm I. Since there will be more unused straight arms, you can practice trimming with them first.

An alternative method to trimming is using a half-burned knife to cut the plastic parts off. Leave a little bit longer because melted plastic will have a rounded edge.

> ⓘ Pay attention to the width difference between servo arm I's two long sides, as well as the trimming location (using screw holes as references).

As we switch servo suppliers, the servo arms also changed a little bit (as shown below). To reduce confusion, the servo arm I can be used for most of the joints, just trim it accordingly. Servo arm K is used for the tail. Servo J can be omitted.

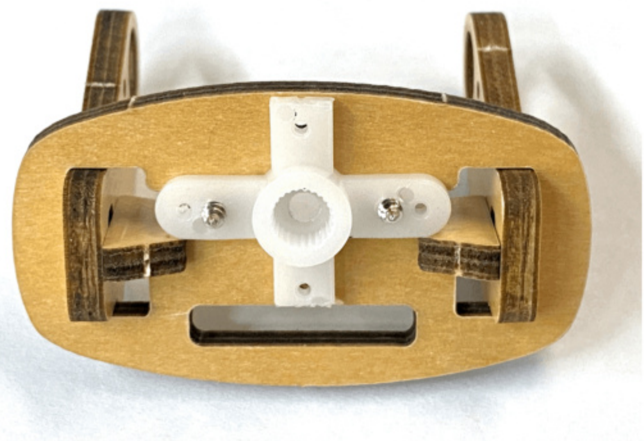


> ⓘ In kits after January 2021, we included the new servo arms that requires minimal efforts to cut. You

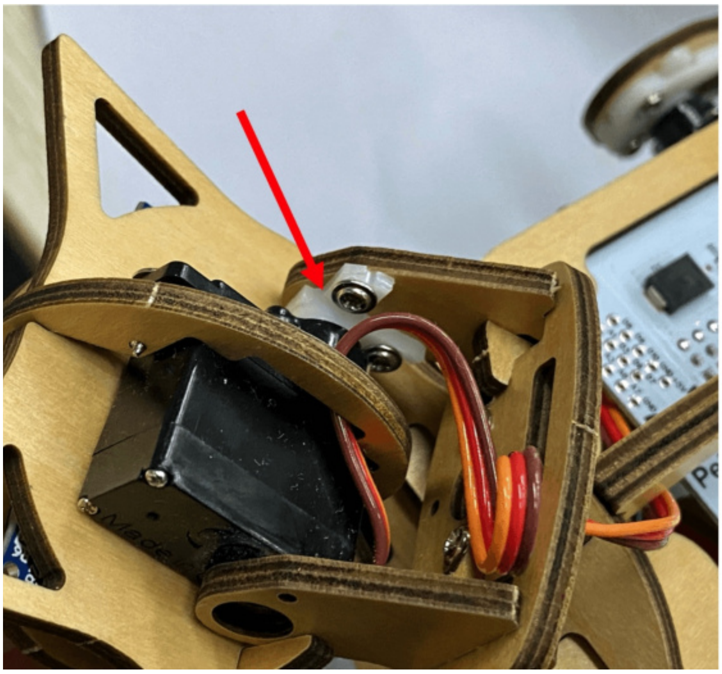
> only need to cut the cross-shaped servo arm to make the L-shaped connector for the neck joint.

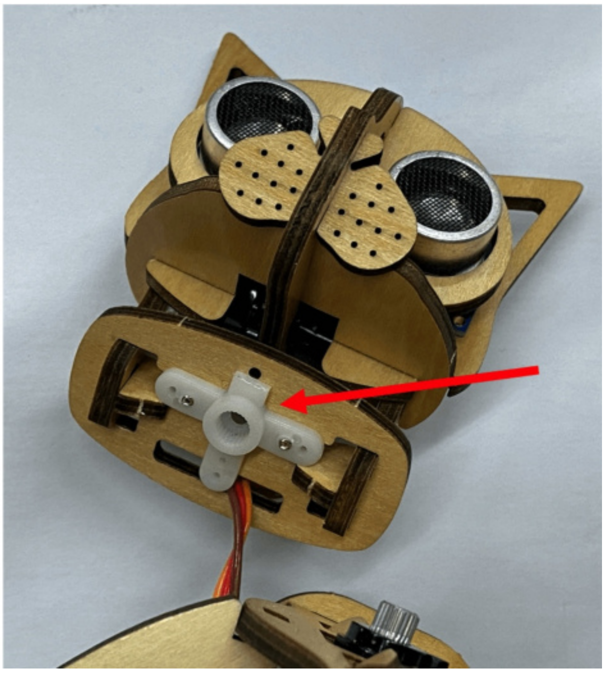
3.1.4. Assemble the head group.

Note that the base should only be partially assembled for later calibration. Otherwise, it will be difficult to insert the servo between neckpieces. Also, notice how the servo wire is organized in the head. Assemble the head group as shown in the head animation.

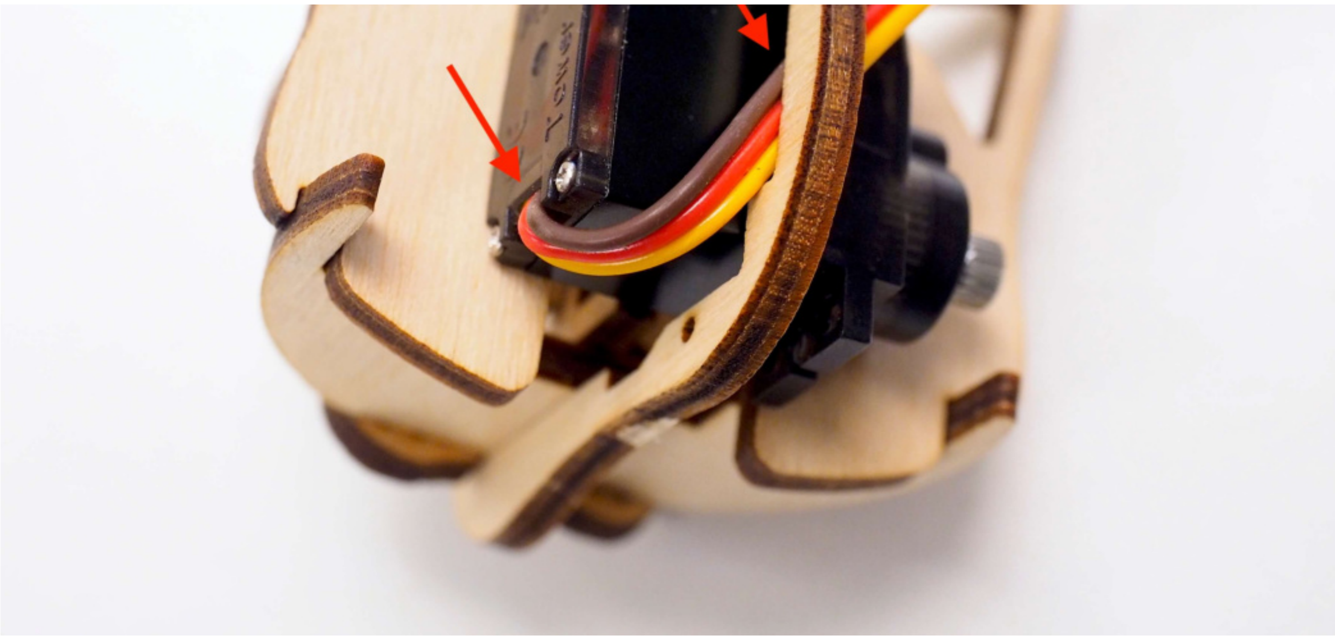












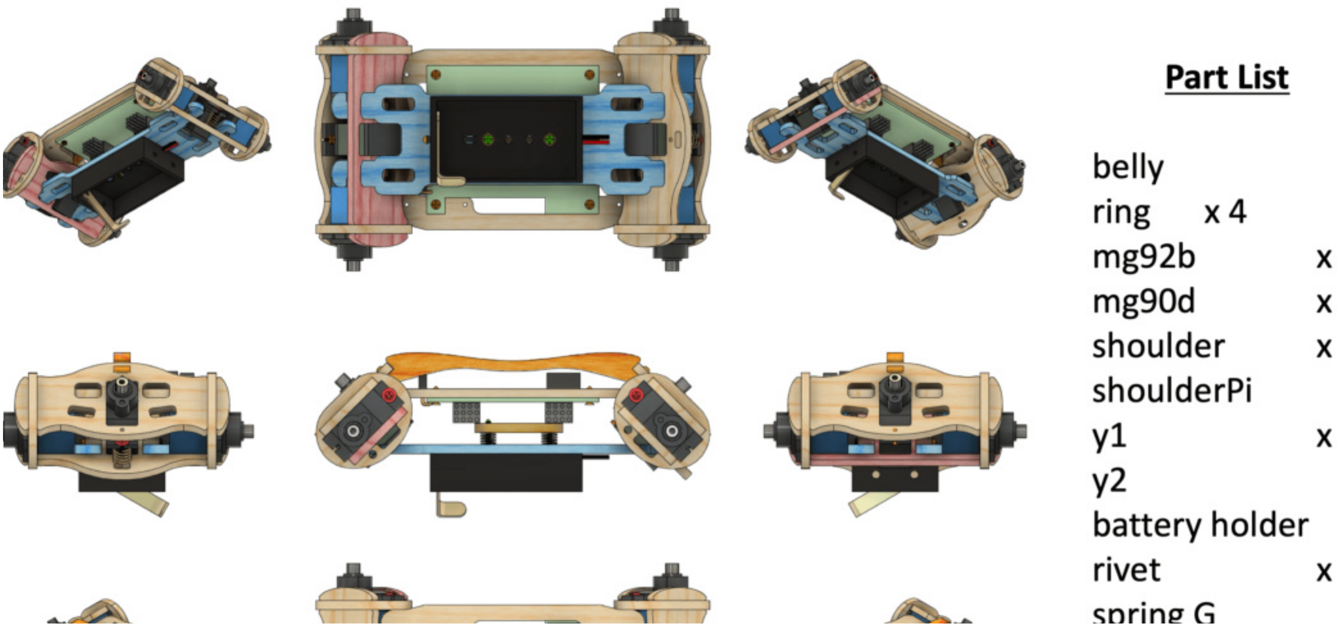
> ⓘ DO NOT connect the head with the neck yet, because the tilt servo on the head has to be calibrated.

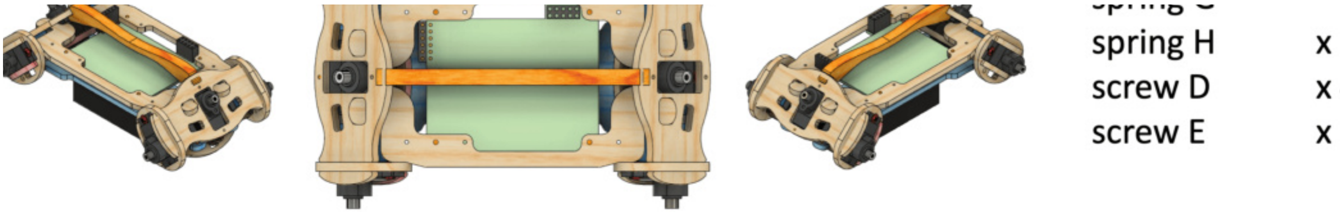
3.2. Body

3.2.1. Part list

- NyBoard only

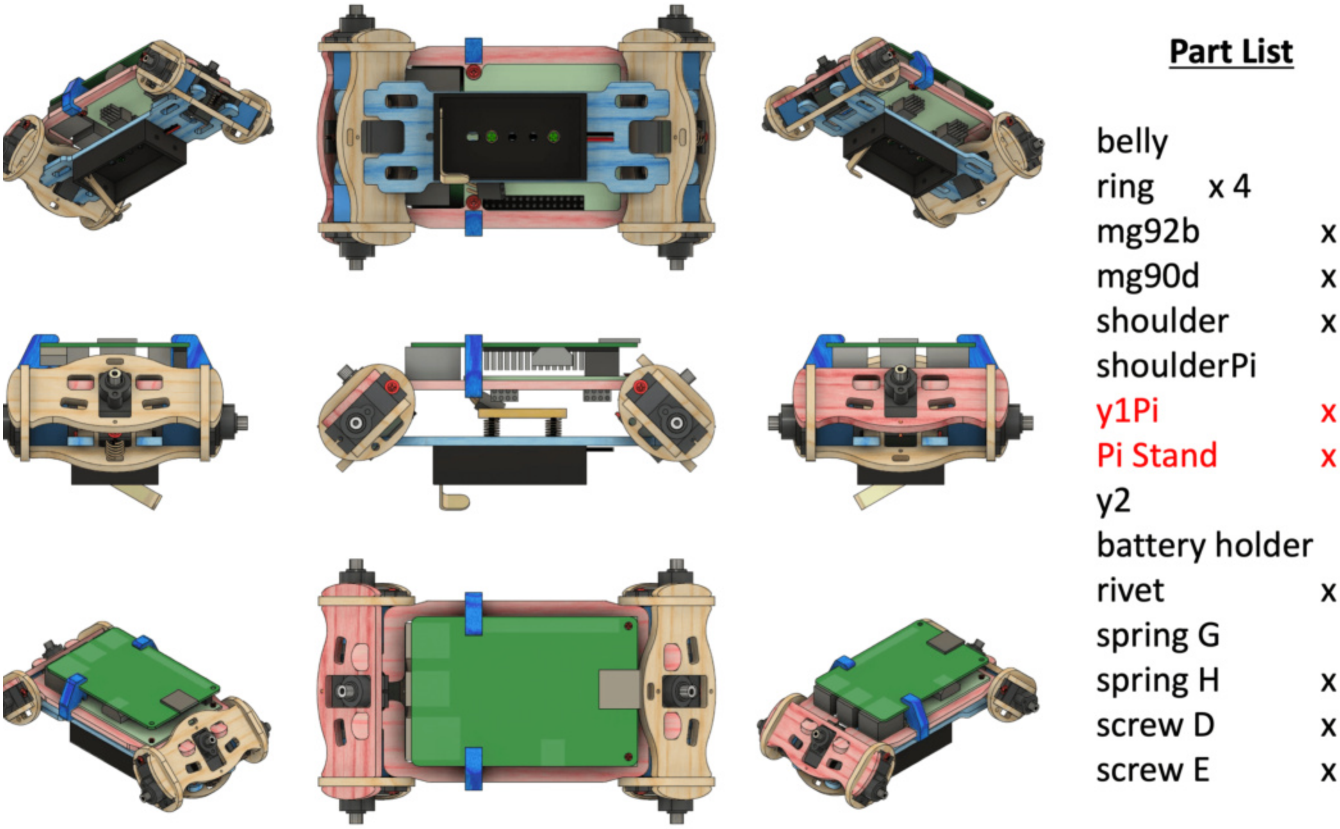
Note that without the Raspberry Pi, the NyBoard is mounted to the underside of y1 with the servo connections facing downward. In later versions, the y1 piece is designed to be symmetric so that both ends have two screw holes.





- NyBoard with Raspberry Pi

Use y1Pi to replace y1, and add Pi Stand. Pay attention to the location of the pink pieces. Note that with the Raspberry Pi, the NyBoard is mounted on top of y1Pi.



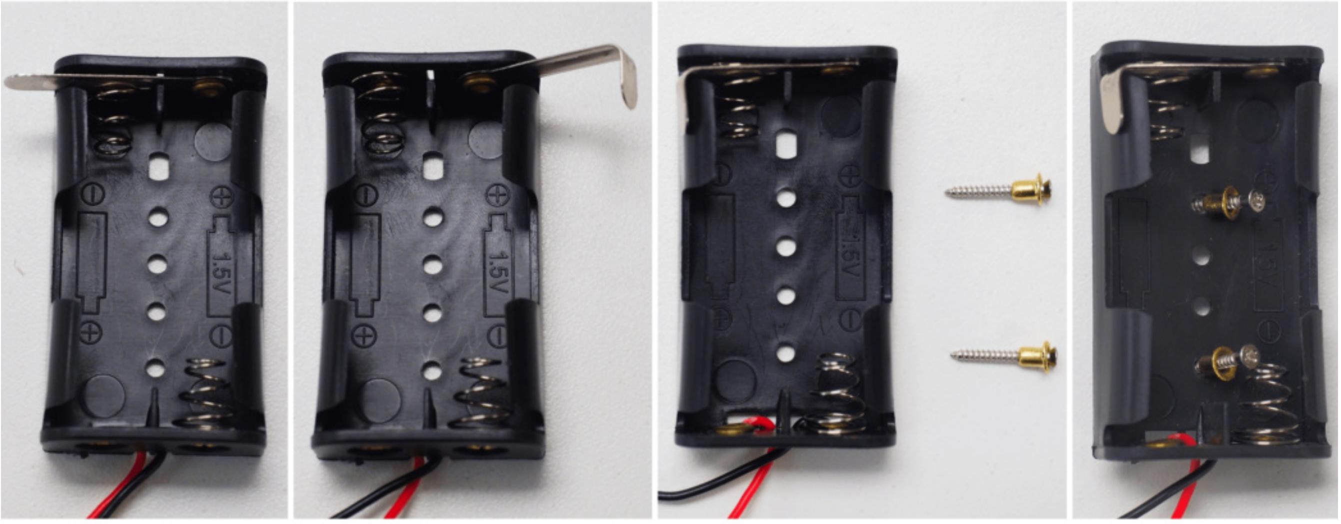
> On earlier batches of NyBoards, the manufacturer used taller jumper pins than expected, so it will be necessary to bend pins or otherwise modify the NyBoard to fit a Pi on top and use the Pi Stand. There is a list of suggested solutions in the forum by this link: https://www.petoi.com/forum/clinic/placement-of-raspberry-pi-3b

- Other controllers

I also included 5 x 1”/4 nuts for mounting other circuit boards.

3.2.2. Install the adjustable battery holder to the belly

Bend the hinge L of the battery holder to 90 degrees, close to the wall. It functions as a switch. Insert the long screw E through the rivet so that you can better handle the rivet. Insert and push the rivet into the hole on the bottom of the battery holder. Pay attention to the holes’ locations.

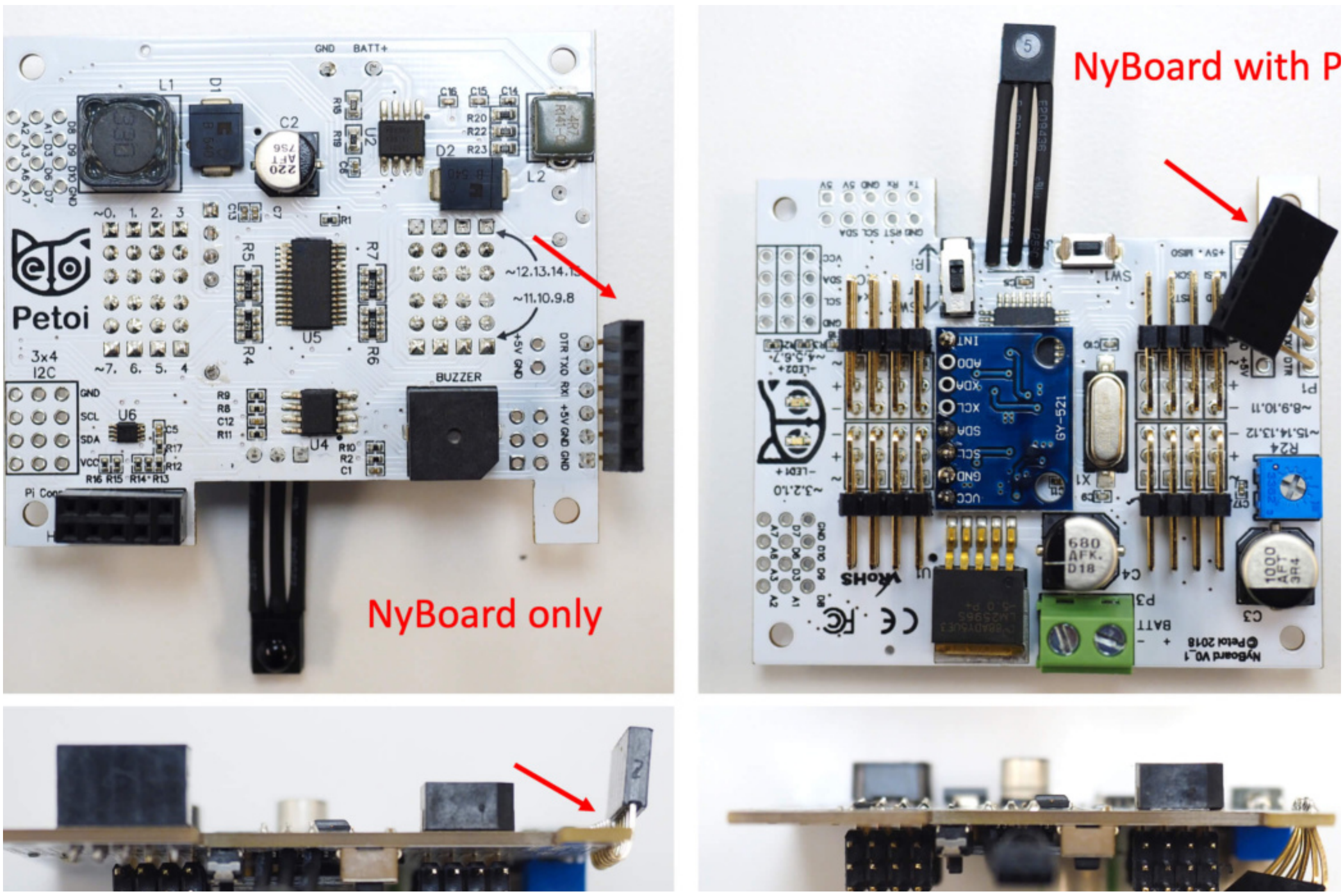


The spring attached structure of the battery holder is used for shifting the center of mass when fine-tuning gaits.

> ⓘ The battery holder is generic for AA (1.5V) batteries. But Nybble uses 3.7V Li-ion batteries.

3.2.3. Assemble the body group

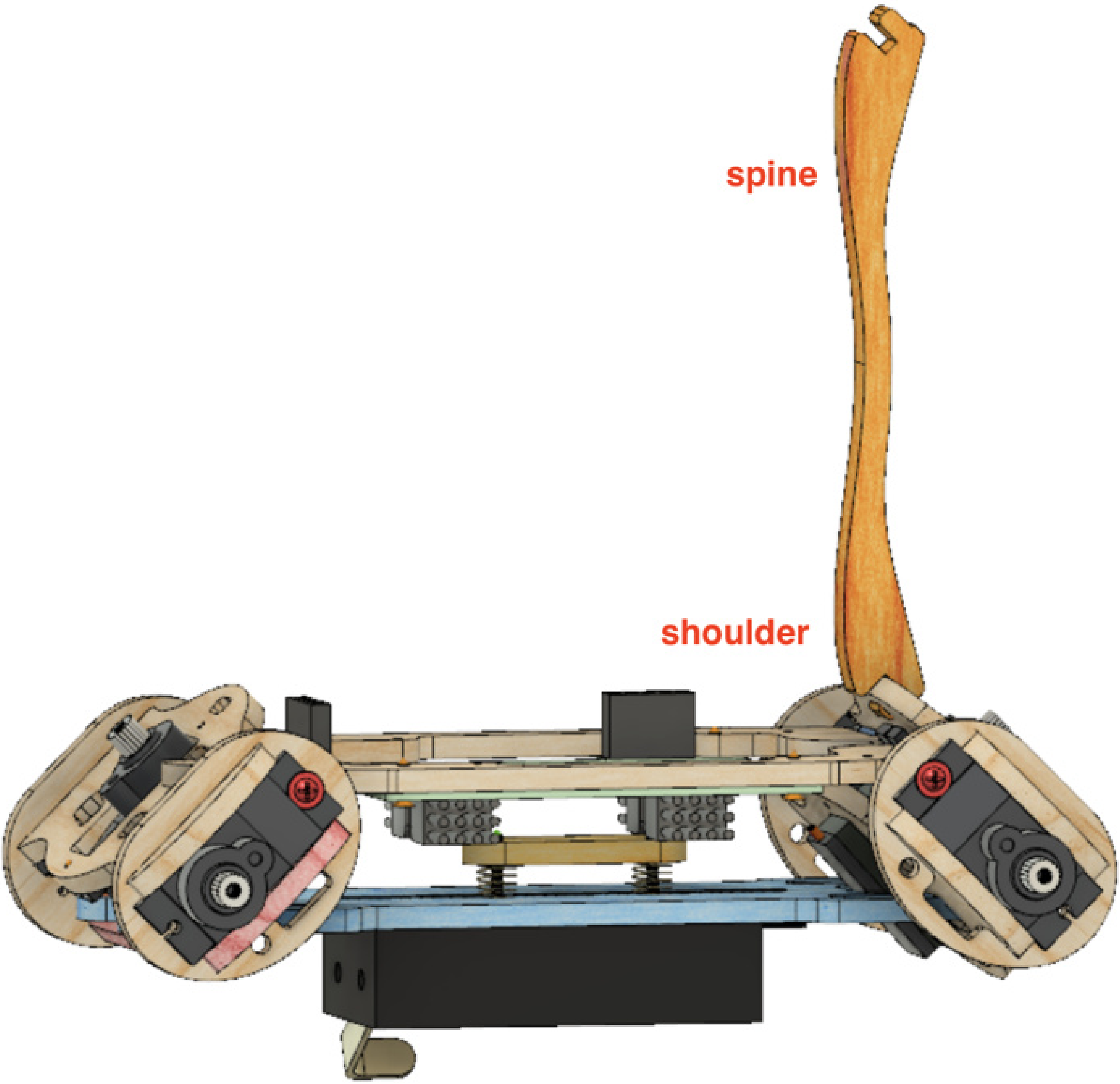
Pay attention to the long pins of the infrared receiver and FTDI port. They are designed to be bent in favorable directions. Don’t bend the pins too often or it will lead to metal fatigue. Observe the adjusted configuration if you want to mount a Pi.





Assemble the body group as shown in the body animation.

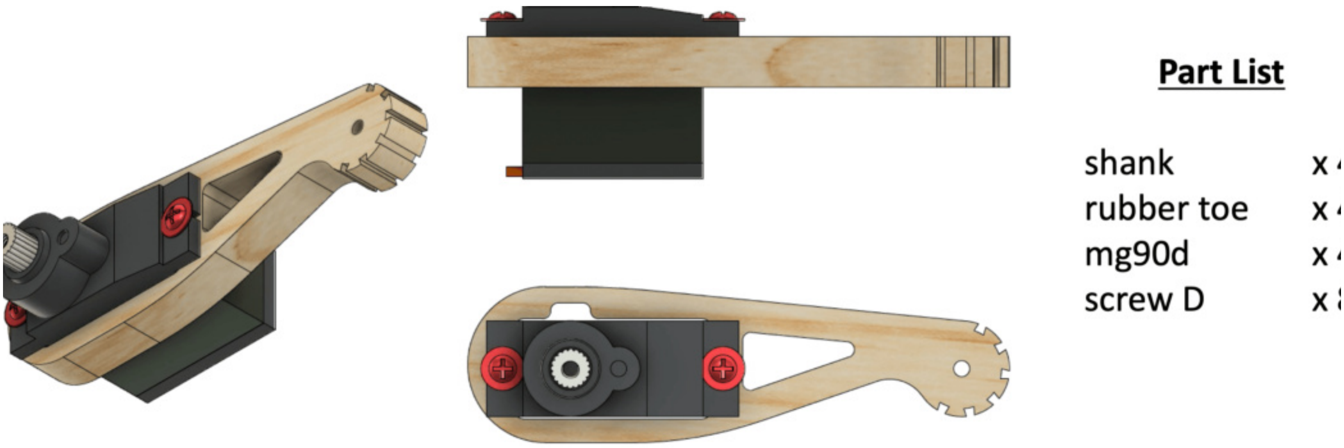
ⓘ The spine piece may be thicker than the slot on the shoulder. You can insert it from the outside first to compress the tip and widen the slot, then insert it to the inside of the shoulder.



3.3. Shank

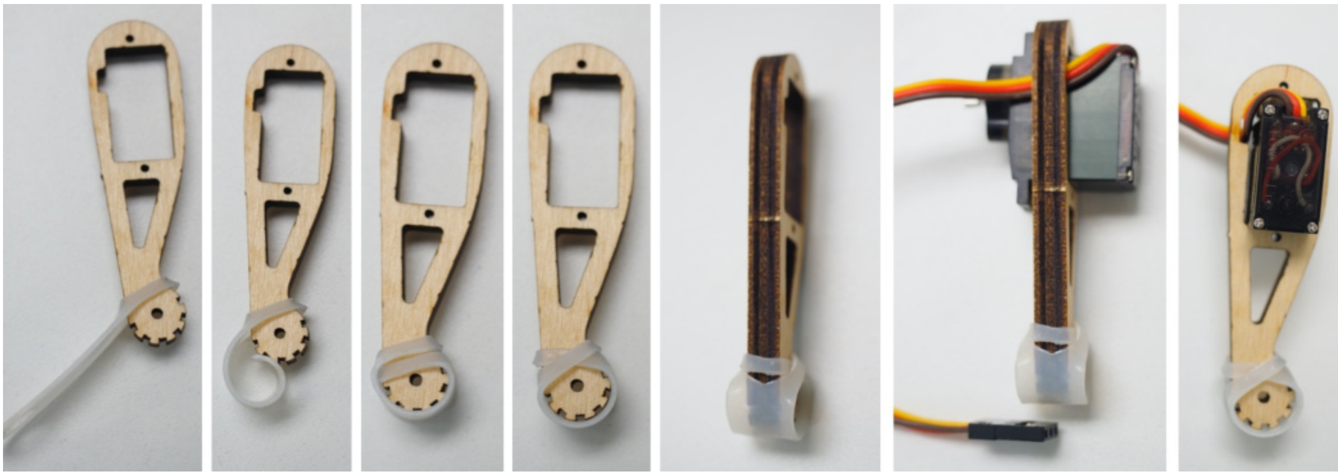
3.3.1. Part list





3.3.2. Attach the rubber to the tip of the shank.

The serrated structure on the tip of the shank is already good for walking. The rubber toe is optional to increase friction and soften each step.



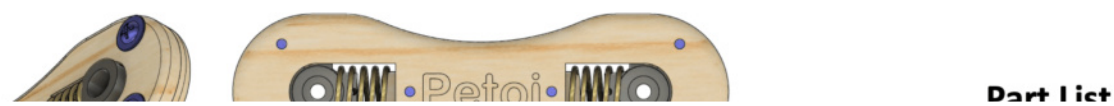
3.3.3. Insert the servo into the slot on the shank.

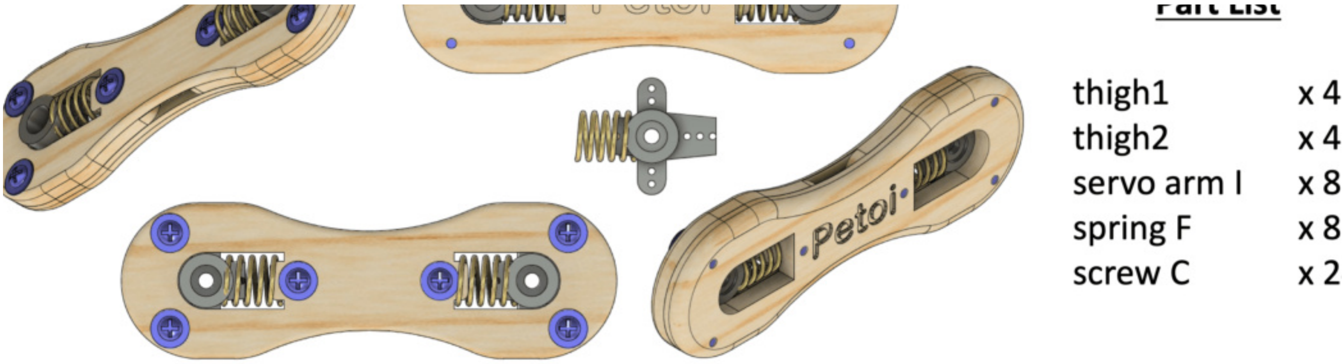
Pay attention to the direction that the wire is twisted. The small dent on the long edge is designed to let the wire go through. Think about the symmetry of the four legs. Assemble the shank as shown in the shank animation.

> ⓘ DO NOT install the servo screw A yet.

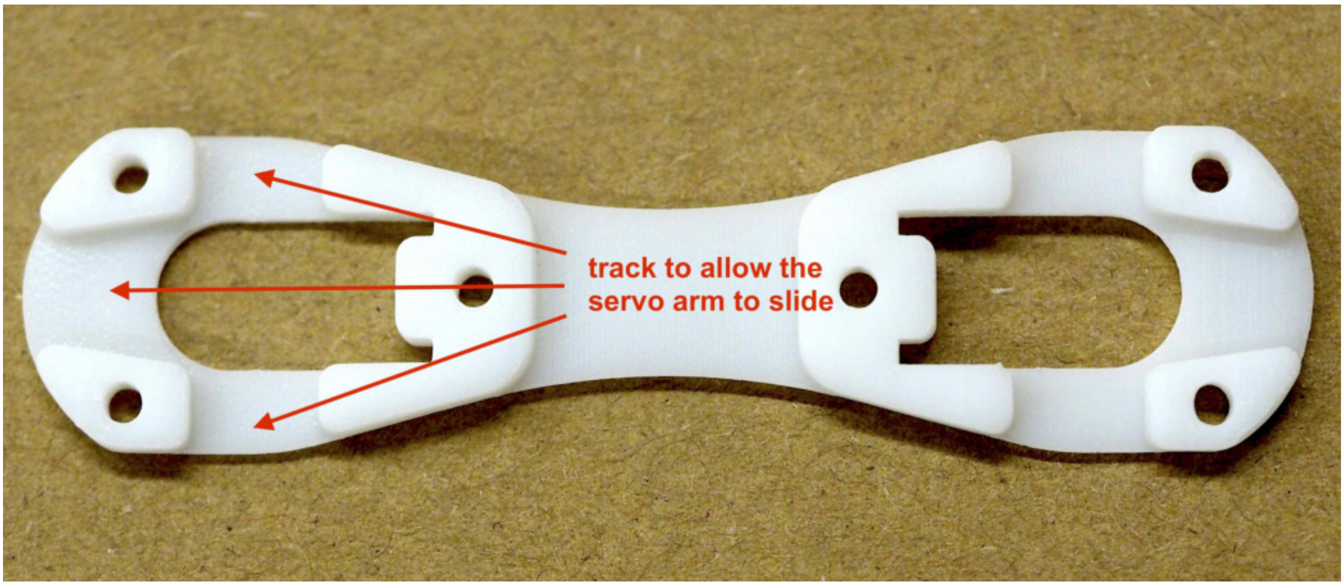
3.4. Thigh

3.4.1. Part list





In later versions, we will use plastic pieces for thigh2.

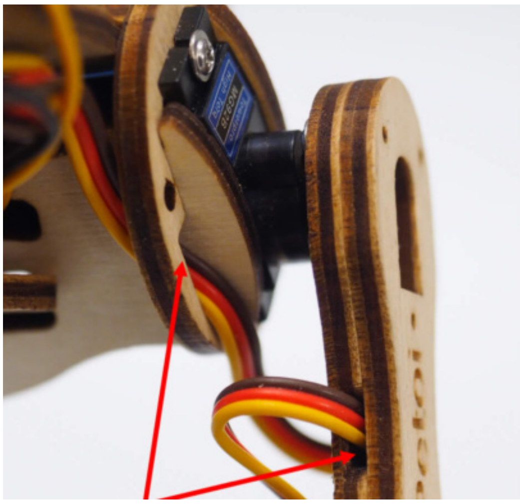


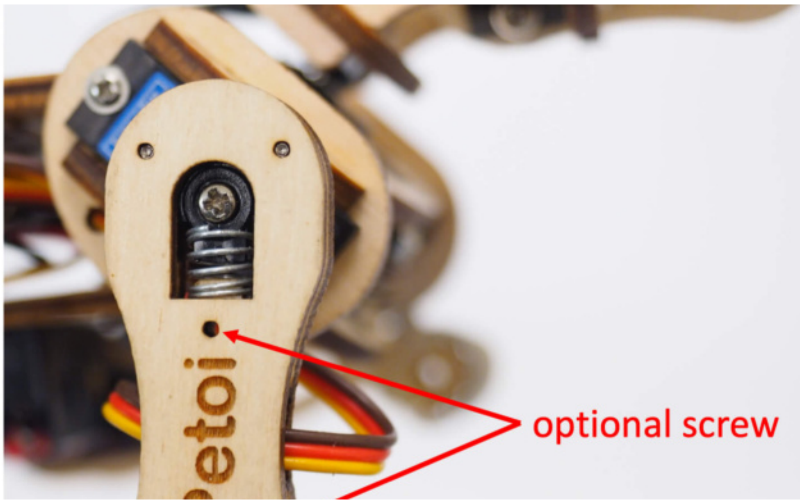
3.4.2. Trim the servo arms for attaching servos.

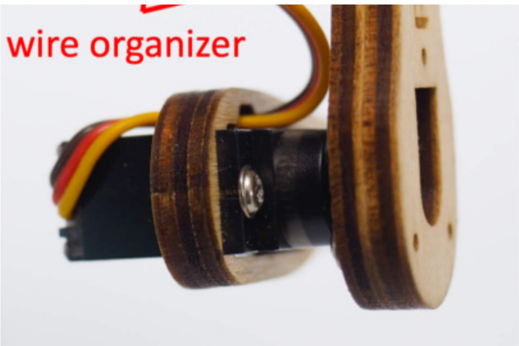
The location has been shown in the Head and Neck section. The trimmed narrower servo arm is designed to be inserted into spring F.

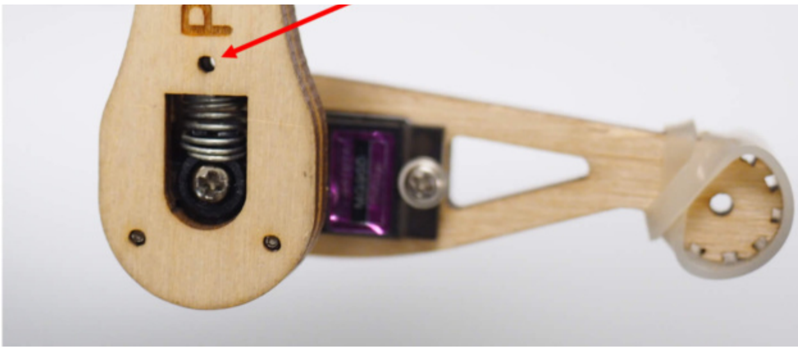
3.4.3. Assemble the thigh.

Before closing thigh1 and thigh2, put the wire of the shank through the slot in the middle of the thigh. Think about the symmetry of the four legs.









The servo arm should be able to slide in the track on thigh2 with subtle friction after thigh1 and thigh2 are screwed together. You can tune the tightness of screw C to achieve proper friction. If you need more control over the tightness:

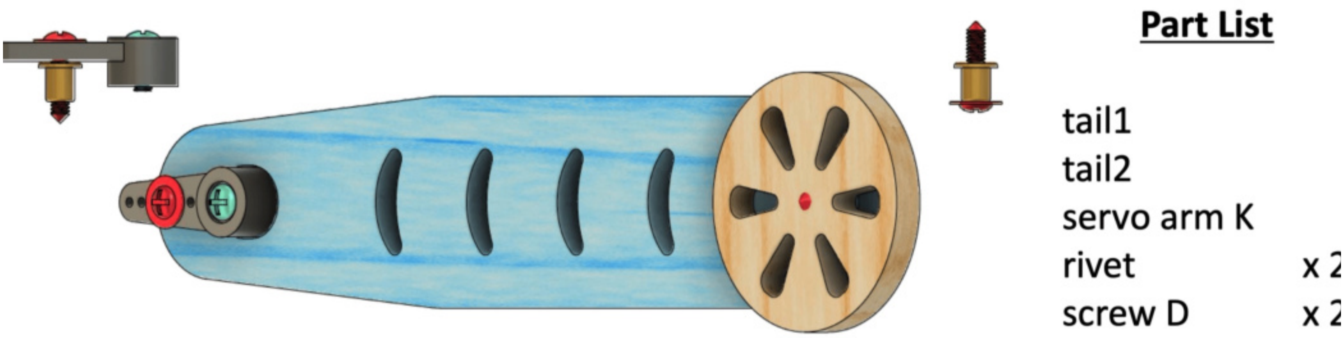
- Scratch the track using a flat screwdriver to reduce friction.
- Apply a little paper glue to the track and let dry to increase friction.

Assemble the thigh as shown in the thigh animation.

> ⓘ DO NOT screw neck and legs to the body's servos yet.

3.5. Tail

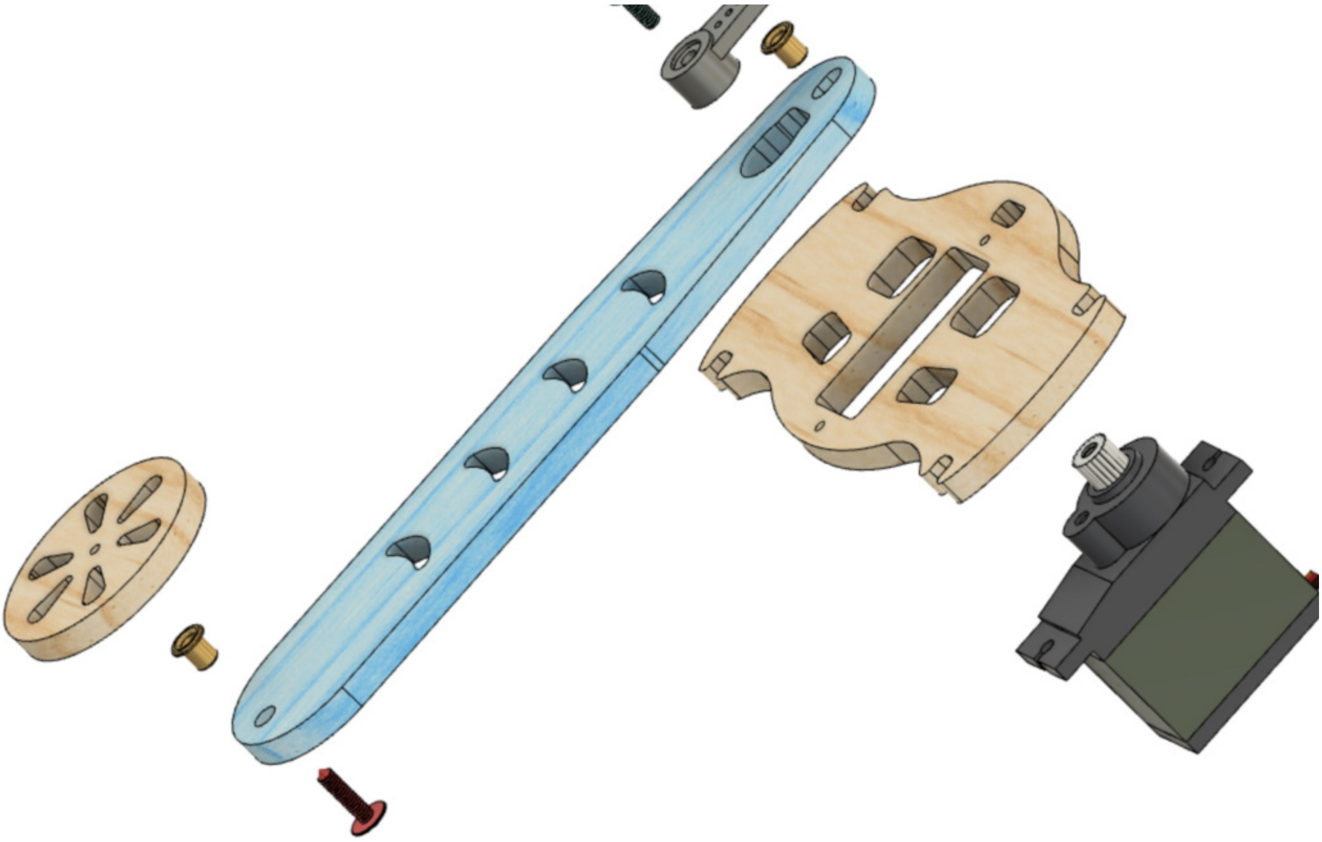
3.5.1. Part list



3.5.2. Assemble the tail.

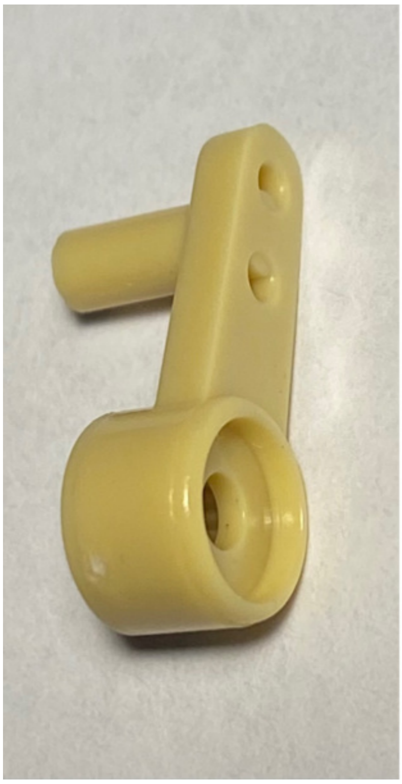
The screw D is installed in the third hole counted from the center of the servo arm K. Pay attention to the order that every piece is stacked. The wheel (tail2) should be able to rotate with little friction, and the whole tail should be able to tilt by a small degree.





Assemble the tail as shown in the tail animation.

ⓘ In kits after January, we inlcuded a special servo arm to simplify the installation of the tail joint.

ⓘ DO NOT connect the tail to the body yet.

4 ⌨️ Configuration

"Who loves fried chips?" 🍟

The following video shows the software configuration using our robot dog Bittle. The steps are identical to Nybble except that you need to change the model definition in OpenCat.h to Nybble.

```
//#include "InstinctBittle.h" //activate the correct header file according to your model
#include "InstinctNybble.h"
```



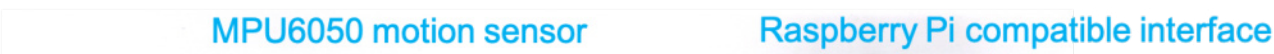
Quick Software Setup Guide

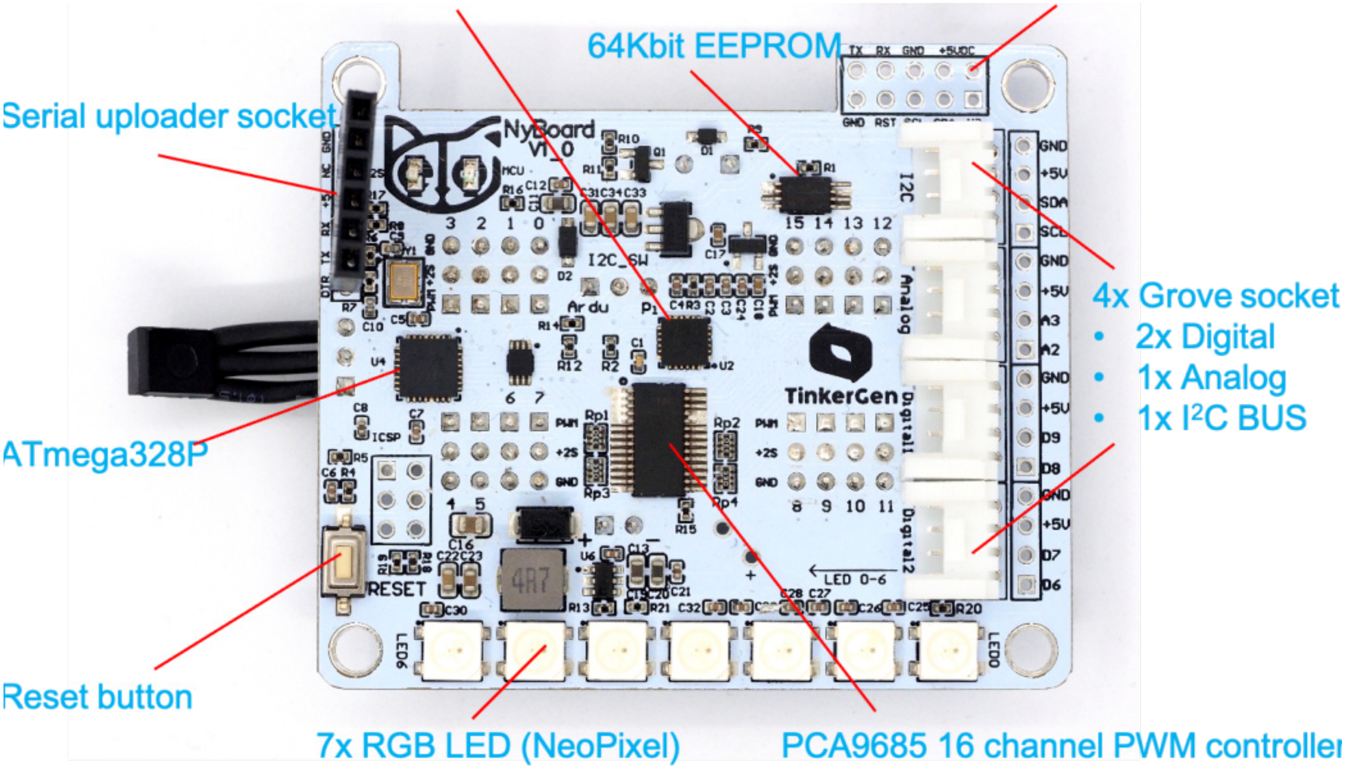
4.1. NyBoard

4.1.1. Read the user manual

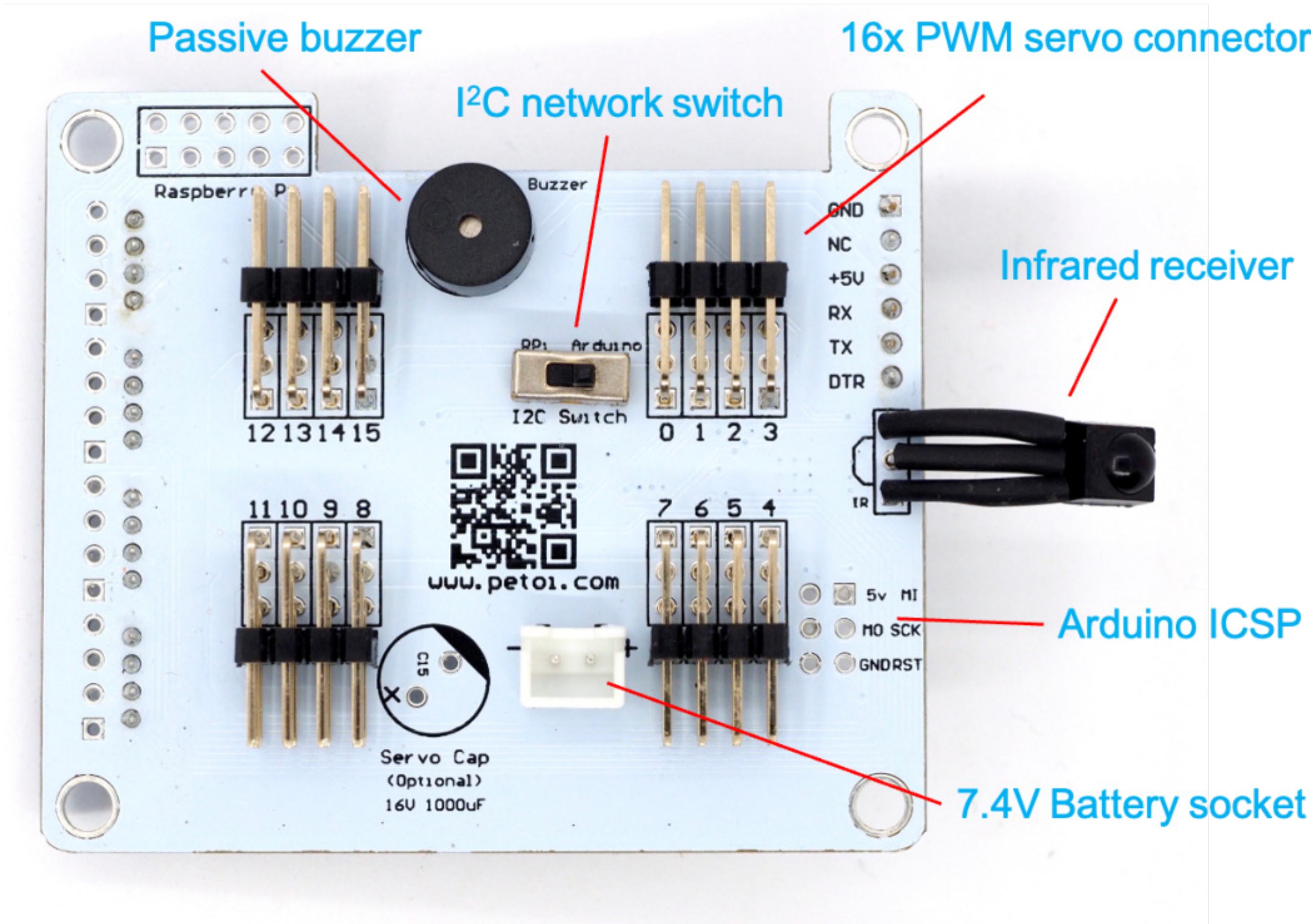
Find the version info on NyBoard. Read the user manual for NyBoard V0_1, NyBoard V0_2 , NyBoard V1_0, or NyBoard V1_1 (a light revision) accordingly.

> ⚠ Wrong operations may damage your NyBoard!





NyBoard V1_0 Top side



NyBoard V1_0 Bottom side

4.1.2. Dial the I2C switch (SW2) to Ar.

The I2C switch changes the master of I2C devices (gyro/accelerometer, servo driver, external EEPROM). On default “Ar”, NyBoard uses the onboard ATmega328P as the master chip; On “Pi”, NyBoard uses external chips connected through the I2C ports (SDA, SCL) as the master chip.

> Sometimes if you cannot go through the bootup stage, maybe you have accidentally dialed the switch to "Pi".

> The following section is kept for older versions (V0)
>
> **4.1.3. Dial the potentiometer clockwise to start from the lowest voltage.**
>
> > From NyBoard V0_2, we provide a jumper selector to bypass the potentiometer. If you are using the metal servos in a standard Nybble kit, this section can be skipped.
>
> Higher voltage will increase the servos’ torque and make Nybble move faster. The downside is it will increase current draw, reduce battery life, affect the stability of the circuit, and accelerate the wearing of the servos. Based on my tests, 5.5V seems to result in a balanced performance.
>
> For the initial installation, don’t put screws on the NyBoard as you may need to take it out for tuning the potentiometer. Make sure all the servos can rotate fine in normal working conditions before making fine calibrations.
>
> **4.1.4. Adjust the NyBoard for optimized performance (no need for NyBoard V1_0).**
>
> NyBoard is designed for two use cases. One is for Nybble that uses metal-geared servos, the other is for DIY robots that may use plastic geared servos. Plastic servos can only stand 6V so there is a step-down chip on NyBoard. The chip is rated for 5A maximal output, but can only be achieved with multiple proper settings and careful tuning.
>
> When using NyBoard with the metal-geared servos of Nybble, optimized performance can be achieved by some adjustment. For NyBoard_V0_1, you will need to do some soldering work as discussed in the forum post. For NyBoard_V0_2, you can connect the jumper switch SW3 between BATT and V_S (Considering safety for plastic servos, by default NyBoard comes with SW3 connecting V_S and V+).
>
> > V_S means power for the servo. The jumper switch chooses whether to power the servos (V_S) by the step-down chip (V+) or by the batteries (BATT) directly. So BATT and V+ should never be connected directly.
>
> > It turns out that NyBoard works more stable when powering the metal-geared servos directly with BATT rather than V+. However, if you are using NyBoard to drive your own plastic geared servos, you do need to use the step-down circuit.

4.2. Downloads and installations

ⓘ You will need the newest Arduino IDE to set up the environment. Older versions tend to compile larger hex files that may exceed the memory limit.

If you have previously added other libraries and see an error message "XXX library is already installed", I would recommend you delete them first (instruction: https://stackoverflow.com/questions/16752806/how-do-i-remove-a-library-from-the-arduino-environment). Due to different configurations of your Arduino IDE installation, if you see any error messages regarding missing libraries during later compiling, just google and install them to your IDE.

ⓘ If you downloaded the newest OpenCat code from GitHub after Jan 3, 2022, you can skip all the following library.

4.2.1. Install through the library manager

Go to the library manager of Arduino IDE (instruction: https://www.arduino.cc/en/Guide/Libraries), search and install

- **Adafruit PWM Servo Driver**
- **QList (optional)**
- **IRremote**

The IRremote library was updated recently. And for some reason, they even changed the encoding of the buttons. You **MUST** roll back the library's version to **2.6.1** in the library manager.

To save programming space, you **MUST** comment out the unused decoder in IRremote.h. It will save about 10% flash!

Find **Documents/Arduino/libraries/IRremote/src/IRremote.h** and set unused decoders to 0. That is, only **DECODE_NEC** and **DECODE_HASH** are set to 1, and others are set to 0.

```
#define DECODE_RC5           0
#define SEND_RC5             0

#define DECODE_RC6           0
#define SEND_RC6             0

#define DECODE_NEC           1
#define SEND_NEC             0

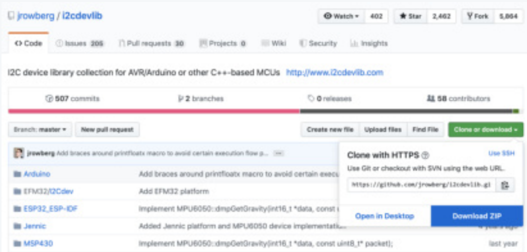
#define DECODE_SONY          0
#define SEND_SONY            0

...
set zeros all the way down the list
...

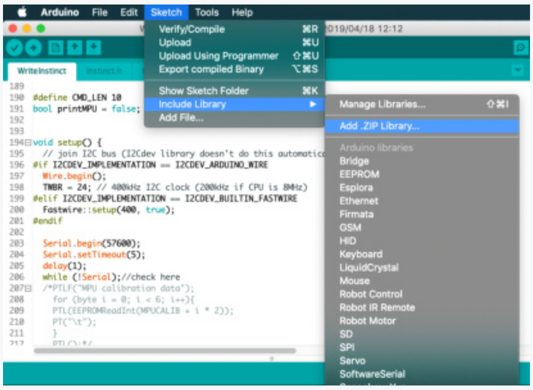
#define DECODE_HASH          1 // special decoder for all protocols
```

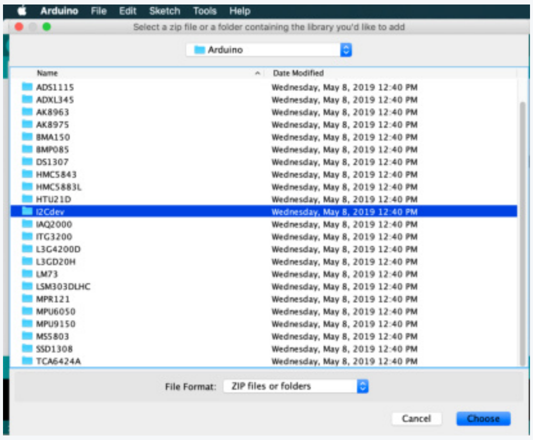
Install by adding .ZIP library

Go to jrowberg/i2cdevlib: I2C device library collection for AVR/Arduino or other C++-based MCUs, download the zip file, and unzip. You can also git clone the whole repository.



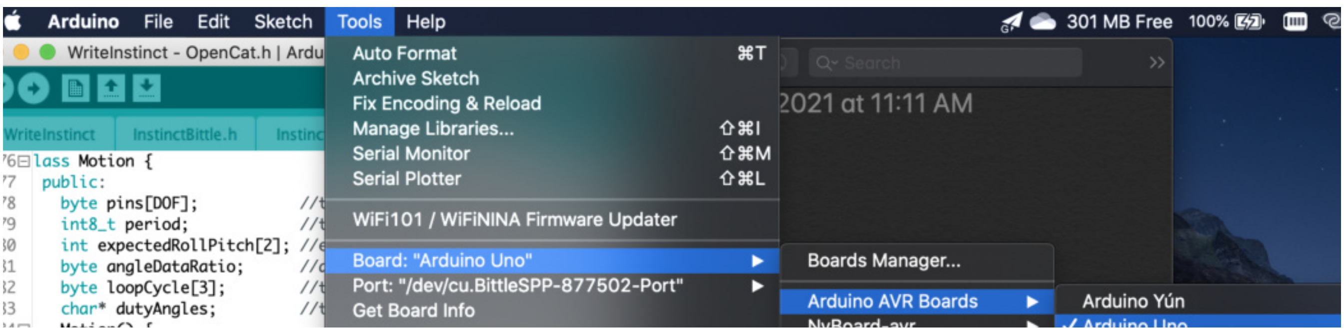
Use **Add .ZIP Library** to find **Arduino/MPU6050/** and **Arduino/I2Cdev/**. Click on the folders and **add them one by one**. They don't have to be **.ZIP** files.

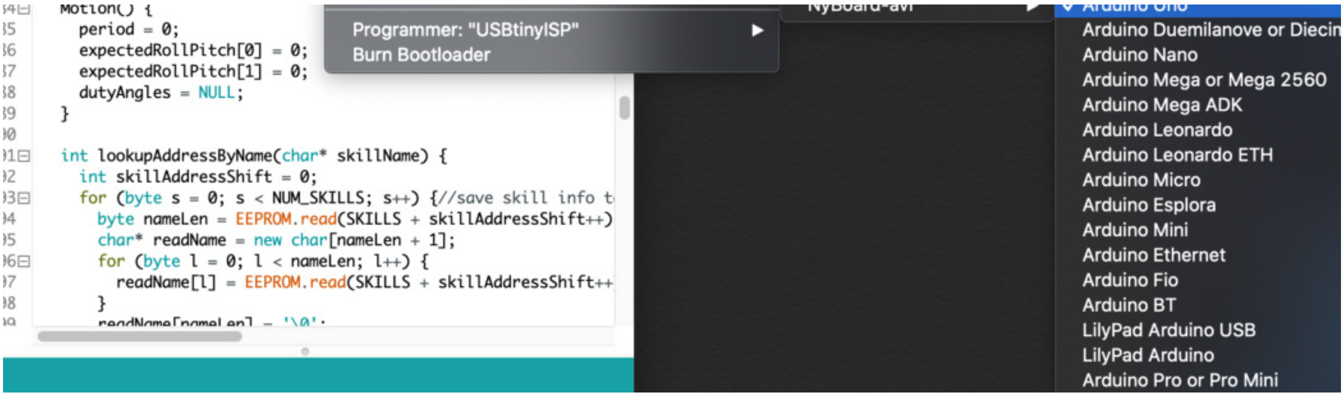




4.2.3. Add NyBoard support to Arduino IDE

- If you get **NyBoard V1_***, you can simply choose **Arduino Uno**.

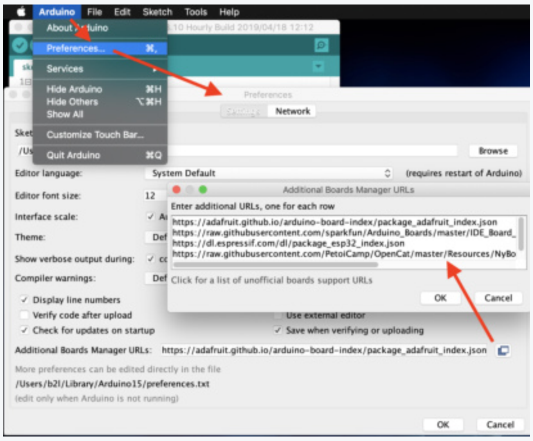




ⓘ The following section is kept for older versions (V0)

- If you have **NyBoard V0_*** which runs at 20MHz, you need to add our customized configuration file following the next steps:

Automatic method (many thanks to A-Ron!)



1. Open the **Preferences...** panel
2. In the **Additional Boards Manager URLs** field, add **https://raw.githubusercontent.com/PetoiCamp/OpenCat/master/Resources/NyBoard/boardManager/package_petoi_nyboard_index.json**. If there are already URLs listed in this field, separate them with commas or click on the icon next to the field to open up an editor and add this URL to a new line. Make sure there's no space in the address when you copy-paste the link.
3. Click on the OK button to close the Preferences panel (you have to close the previously opened **Additional Boards Manager URLs** editor first, to close the Preference panel)
4. Open the **Boards Manager...** window with the **Tools -> Board: XXXX -> Boards Manager...** menu item.
5. In the **Filter your search...** field, type **NyBoard**
6. Select the entry and click on **Install**
7. Click on the **Close** button
8. Select **ATmega328P (5V, 20 MHz) NyBoard** from the **Tools -> Board: XXXX** menu (NyBoardV0_1 and NyBoardV0_2 are using the same board settings.)

Manual Board Installation Method

> ⓘ Only if the above method fails

- Locate the file **boards.txt**

Mac location:

/Users/UserName/Library/Arduino15/packages/arduino/hardware/avr/version#/

Or:

/Applications/Arduino.app/Contents/Java/hardware/arduino/avr

To access, right-click on Arduino.app and choose Show Package Contents

Windows location:

C:\Program Files(x86)\Arduino\hardware\arduino\avr\

IMPORTANT! If you have installed the Arduino IDE via the Microsoft Store, you likely will not have access to the folder where critical configuration files are stored. The easiest solution is to uninstall the IDE and download/re-install it directly from https://www.arduino.cc.

Linux

Downloading from the terminal or from the software manager might not give you the latest version which can be an issue. Please download the latest version from Arduino's site: https://www.arduino.cc/en/Main/Software

Unzip the package and `sudo install.sh`

The location of boards.txt files is:

Fedora: boards.txt is symlinked under:

/etc

Arch: boards.txt is found at:

/usr/share/arduino/hardware/archlinix-arduino/avr/

Mint:

location_of_installation/arduino/hardware/arduino/avr

Ubuntu (on 18.04 when installing with `apt-get install arduino`):

/usr/share/arduino/hardware/arduino/boards.txt

after locating **boards.txt**:

- Make a copy of **boards.txt** in case you want to roll back.

- Create new **boards.txt**.

You can download my **boards.txt** file, or:

Edit your **boards.txt** with admin privilege. Find the section of

pro.name=Arduino Pro or Pro Mini

and insert the

```
## Arduino Pro or Pro Mini (5V, 20 MHz) w/ ATmega328P
## --------------------------------------------------
pro.menu.cpu.20MHzatmega328=ATmega328P (5V, 20 MHz) NyBoard

pro.menu.cpu.20MHzatmega328.upload.maximum_size=30720
```

```
pro.menu.cpu.20MHzatmega328.upload.maximum_data_size=2048
pro.menu.cpu.20MHzatmega328.upload.speed=57600

pro.menu.cpu.20MHzatmega328.bootloader.low_fuses=0xFF
pro.menu.cpu.20MHzatmega328.bootloader.high_fuses=0xDA
pro.menu.cpu.20MHzatmega328.bootloader.extended_fuses=0xFD
pro.menu.cpu.20MHzatmega328.bootloader.file=atmega/ATmega328_20MHz.hex

pro.menu.cpu.20MHzatmega328.build.mcu=atmega328p
pro.menu.cpu.20MHzatmega328.build.f_cpu=20000000L

## Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P
## --------------------------------------------------
```

code block in the Arduino Pro or Pro Mini section as below. Save and quit your editor.

```
##############################################################
pro.name=Arduino Pro or Pro Mini

pro.upload.tool=avrdude
pro.upload.protocol=arduino

pro.bootloader.tool=avrdude
pro.bootloader.unlock_bits=0x3F
pro.bootloader.lock_bits=0x0F

pro.build.board=AVR_PRO
pro.build.core=arduino
pro.build.variant=eightanaloginputs

## Arduino Pro or Pro Mini (5V, 20 MHz) w/ ATmega328P
## --------------------------------------------------
pro.menu.cpu.20MHzatmega328=ATmega328P (5V, 20 MHz) NyBoard

pro.menu.cpu.20MHzatmega328.upload.maximum_size=30720
pro.menu.cpu.20MHzatmega328.upload.maximum_data_size=2048
pro.menu.cpu.20MHzatmega328.upload.speed=57600

pro.menu.cpu.20MHzatmega328.bootloader.low_fuses=0xFF

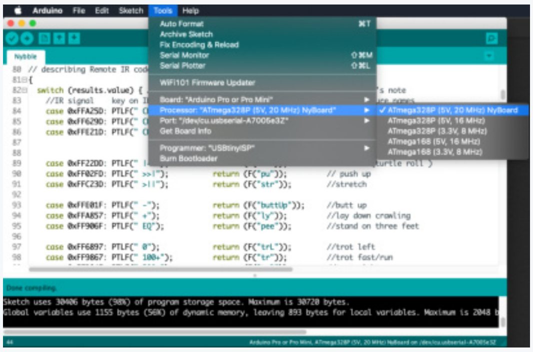
pro.menu.cpu.20MHzatmega328.bootloader.high_fuses=0xDA
pro.menu.cpu.20MHzatmega328.bootloader.extended_fuses=0xFD
pro.menu.cpu.20MHzatmega328.bootloader.file=atmega/ATmega328_20MHz.hex

pro.menu.cpu.20MHzatmega328.build.mcu=atmega328p
pro.menu.cpu.20MHzatmega328.build.f_cpu=20000000L

## Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P
## --------------------------------------------------
```

- Download ATmega328_20MHz.hex and put it in your Arduino folder ./bootloaders/atmega/. You should see other bootloaders with the **.hex** suffix in the same folder.

Restart your Arduino IDE. In **Tools->Boards**, select Arduino Pro or Pro Mini. You should find ATmega328P (5V, **20** MHz) in the Processor menu.



If you cannot find the board, your Arduino IDE may be using the boards.txt in another path. Search boards.txt in all the folders on your computer. Find out the right file that's in effect.

Only if the bootloader of NyBoard collapsed, which is very unlikely to happen

4.2.4. Burn the bootloader

What is a bootloader?

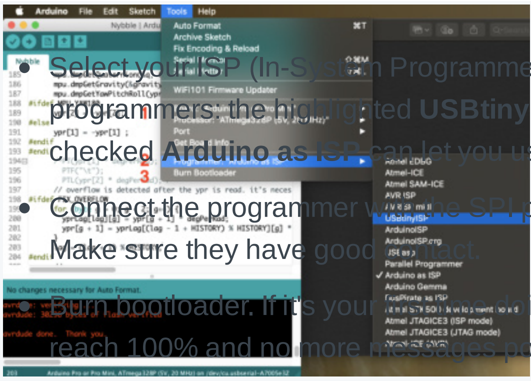
Every NyBoard has to go through functionality checks before shipping, so they should already have a compatible bootloader installed. However, in rare cases, the bootloader may collapse then you won't be able to upload sketches through Arduino IDE.

Well, it's not always the bootloader if you cannot upload your sketch:

- Sometimes your USB board will detect a large current draw from a device and deactivate the whole USB service. You will need to restart your USB service, or even reboot your computers;
- You need to install the driver for the FTDI USB 2.0 to the UART uploader;
- You haven't selected the correct port;
- Bad contacts;
- Bad luck. Tomorrow is another day!

If you really decide to re-burn the bootloader:

- If you get **NyBoard V1_***, you can simply choose **Arduino Uno** under the **Tool** menu of Arduino IDE.
- If you have **NyBoard V0_*** which runs at 20MHz, you need to choose **NyBoard (ATmega328P 5V, 20MHz)**.

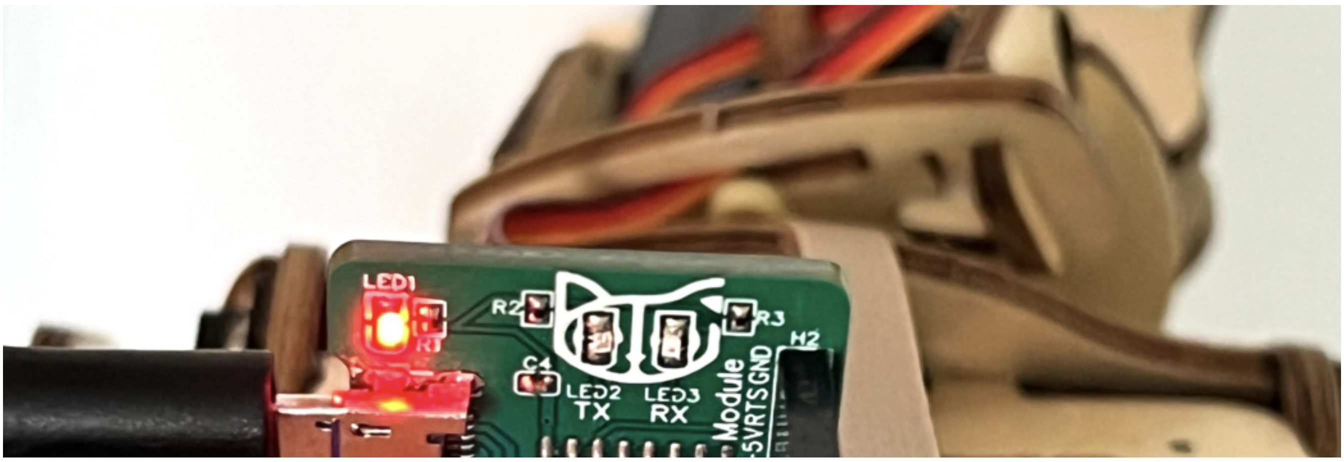


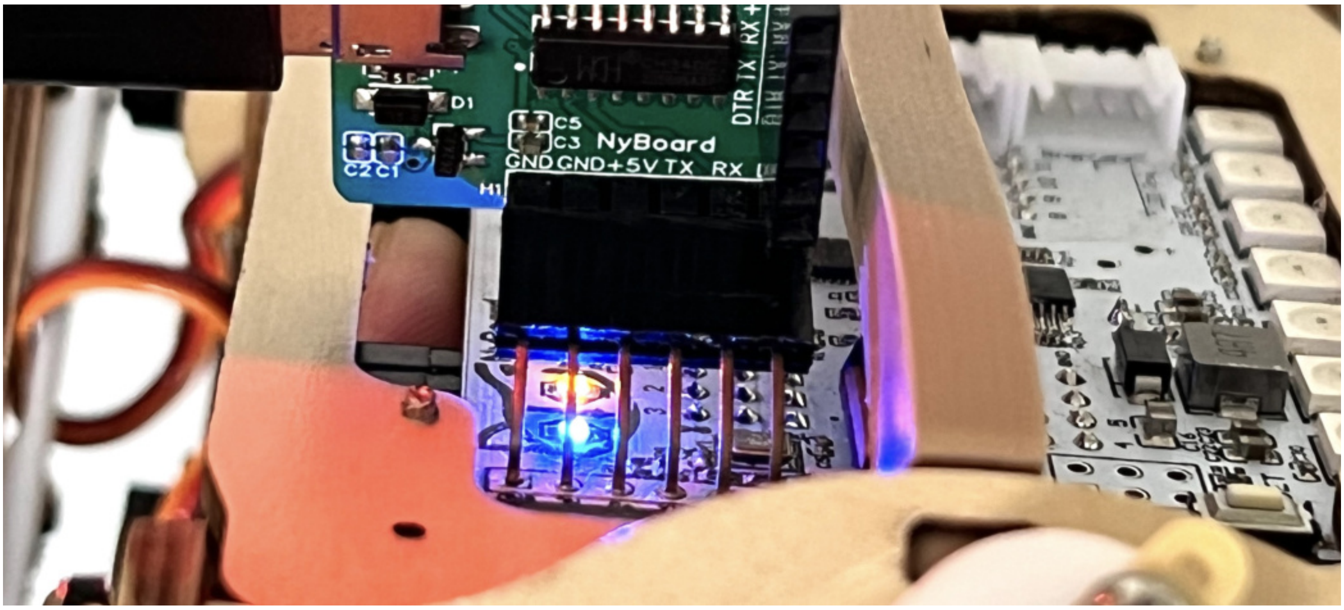
- Select your ISP (In-System Programmer). The above screenshot shows two popular programmers: the highlighted **USBtinyISP** is a cheap bootloader you can buy, while the checked **Arduino as ISP** can let you use a regular Arduino as ISP!
- Connect the programmer with the SPI port on NyBoard. Notice the direction when connecting. Make sure they have good contact.
- Burn bootloader. If it's your first time doing so, wait patiently until you see several percent bars reach 100% and no more messages pop up for one minute.

4.2.5. Connect FTDI uploader

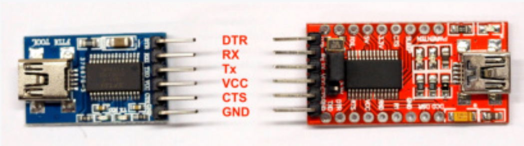
Check the pin marks of the uploader. They should match the marks on the NyBoard.



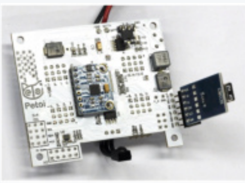
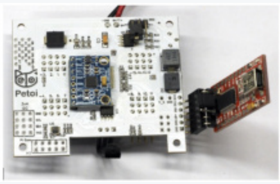




The following picture shows two FTDI boards in previous kits.



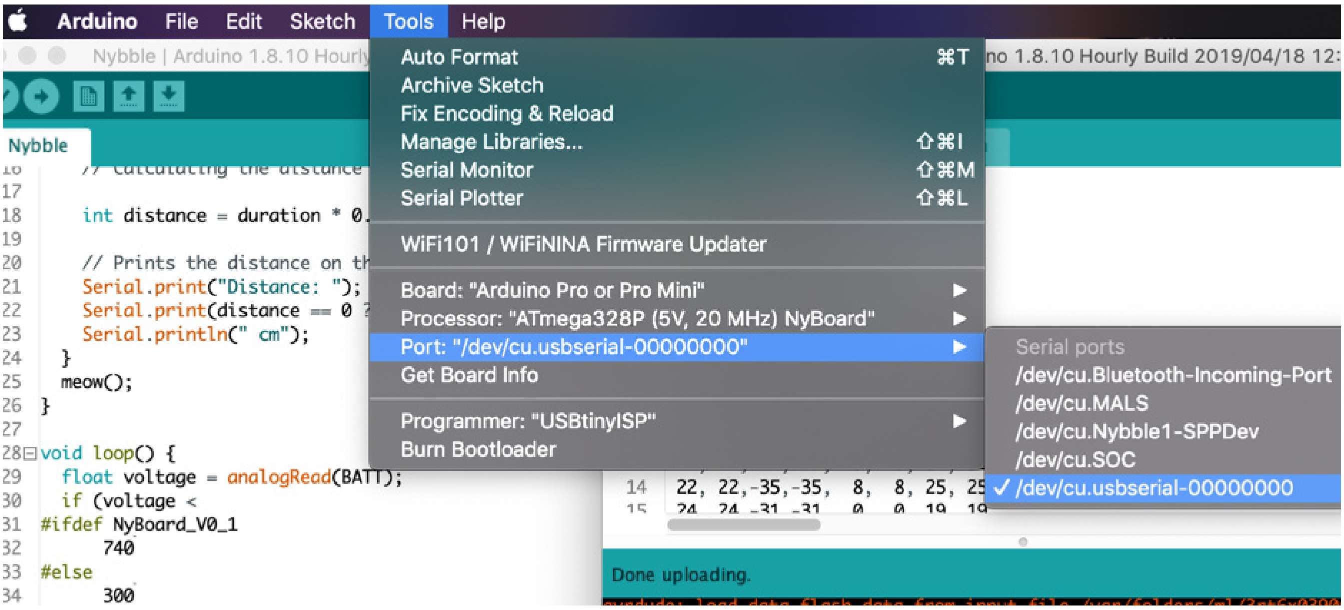
On the red board, the jumper selector should be put on 5V (not 3.3V) on the uploader for Arduino. Match the GND on both the uploader and the 6-pin socket on NyBoard.



Connect your computer with the FTDI uploader through mini-USB to USB cable. The uploader has three LEDs, power, Tx, and Rx. Right after the connection, the Tx and Rx should blink for one second indicating initial communication, then dim. Only the power LED should keep lighting up. You can find a new port under **Tool->Port** as “/dev/cu.usbserial-xxxxxxxx” (Mac) or “COM#” (Windows).

For Linux, once the uploader is connected to your computer, you will see a “**ttyUSB#”** in the serial port list. But you may still get a serial port error when uploading. You will need to give the serial port permission. Please go to this link and follow the instructions: https://playground.arduino.cc/Linux/All/#Permission



If Tx and Rx keep lighting up, there’s something wrong with the USB communication. You won’t see the new port. It’s usually caused by overcurrent protection by your computer, if you’re not connecting NyBoard with an external power supply and the servos move all at once.

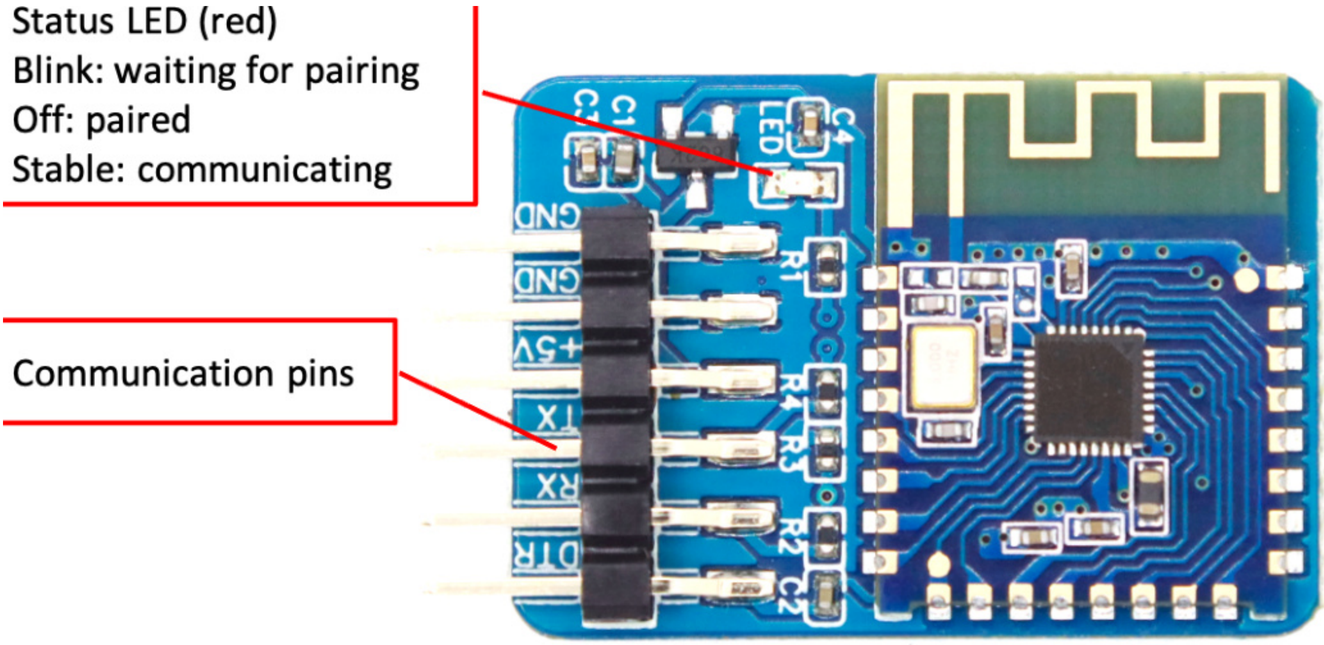
In later versions, there may be a different version of the FTDI uploader. Just check the marks on the chip and connect it with NyBoard correctly.

> ⓘ If you cannot find the serial port after connecting to your computer, you may need to install the driver for the CH340 chip.
>
> - Mac
> - Windows

4.2.6. Connect Bluetooth uploader (optional)

It’s possible to program and communicate with Nybble wirelessly. Check out the Bluetooth instruction on the OpenCat forum. You can even control Nybble with a smartphone APP or web API from there!

Starting from Jan.1st, 2021, we include our official Bluetooth dongle in the standard Nybble kit. It can be used for wirelessly uploading sketches and communicating with the robot. The default passcode is 0000 or 1234, and the baud rate is set to 115200. After connecting to the computer, you can use it in the same way as the wired uploader.



> ⓘ You will need to set its baudrate to **57600** to use it on NyBoard V0_*.
>
> Connect the Bluetooth dongle to the wired uploader (+5V, GND, Rx to Tx, Tx to Rx), and send **AT+BAUD7** through the serial monitor with the setting **Both NL & CR** and baudrate **115200**.

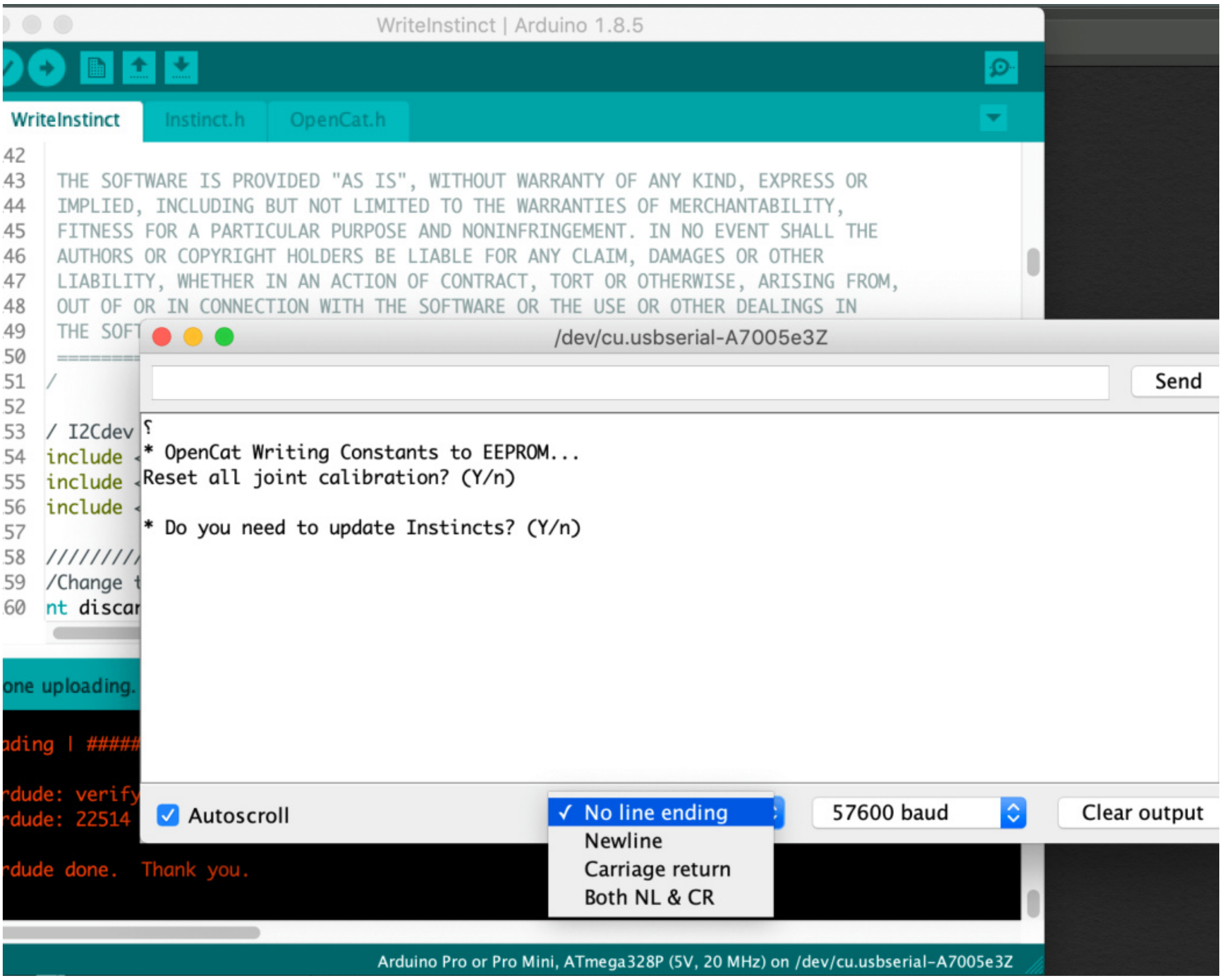
> ⓘ On Mac, the Bluetooth may lose connection after several uploads. In that case, delete the connection and reconnect to resume its functionality.

4.2.7. Download the OpenCat package

- Download a fresh OpenCat repository from GitHub: https://github.com/PetoiCamp/OpenCat. It's better if you utilize GitHub's version control feature. Otherwise, make sure you download the **WHOLE OpenCat FOLDER** every time. All the codes have to be the same version to work together.
- If you download the Zip file of the codes, you will get an **OpenCat-main** folder after unzipping. You need to rename it to **OpenCat** before opening the OpenCat.ino.

> ⓘ No matter where you save the folder, the file structure should be:
>
> ```
> OpenCat/
>         /OpenCat.ino
>         /WriteInstinct/
>                       /WriteInstinct.ino
>                       /OpenCat.h
>                       /InstinctBittle.h
>                       /InstinctNybble.h
> ```

- There are several **testX.ino** codes in **ModuleTests** folder. You can upload them to test certain modules separately. Open any **testX.ino** sketch with prefix “test”. (I recommend using **testBuzzer.ino** as your first test sketch)
- Open up the serial monitor and set up the baudrate:
  - For NyBoard V1_*, choose the board as **Arduino Uno** and later set the baudrate to 115200 in both the code and the serial monitor.
  - For NyBoard V0_*, choose the board as **NyBoard (ATmega328P 5V, 20MHz)** and later set the baudrate to 57600 in both the code and the serial monitor.
- Compile the code. There should be no error messages. Upload the sketch to your board and you should see Tx and Rx LEDs blink rapidly. Once they stop blinking, messages should appear on the serial monitor.
- Make sure you set "**No line ending**" to before entering your commands. Otherwise, the invisible '\n' or '\r' characters will confuse the parsing functions.



For Linux machines, you may see the following error message:

```
rk/exec /home/montagsmodell/Desktop/arduino-1.8.10/hardware/tools/avr/bin/avr-gcc: permission denie
rror compiling for board ATmega328P (5V, 20 MHz) NyBoard V0_1.
```

You will need to add execution privilege to the avr-gcc to compile the Arduino sketch: `sudo chmod +x filePathToTheBinFolder/bin/avr-gcc`

Furthermore, you need to add execution permission to **all files** within /bin, so the command would be : `sudo chmod -R +x /filePathToTheBinFolder/bin`

4.3. Arduino IDE as an interface

With the FTDI to USB converter connecting NyBoard and Arduino IDE, you have the ultimate interface to communicate with NyBoard and change every byte on it.

I have defined a set of the serial communication protocols for NyBoard:

OpenCat
Communication Protocol and Parsing

Interface		Token	Encoding	Parameters		Format	Bytes	Function
asPi erial Port	Arduino Serial Monitor	'h'	Ascii			char	1	print help information
		'c'		idx*,angle**	'\n'	string	strlen + 2	calibrate servo$_{idx}$ by angle
		'm'		idx*,angle**	'\n'	string	strlen + 2	move servo$_{idx}$ to angle
		'j'				char	1	show all 16 joint angles
		'd'				char	1	shut down servos
		'p'				char	1	pause motion
		'a'				char	1	abandon calibration
		's'				char	1	save calibration
		'k'		abbreviation	'\n'	string	strlen + 2	load skill
		'w'		command	'\n'	string	strlen + 2	some future command words
		'r'				char	1	reset board
		'i'	Binary	idx_1 a_1 ... idx_N a_N	'~'	string	strlen + 2	list of indexed rotation angles
		'l'		a_1 a_2 ... a_{DoF}	'~'	string	DoF + 2	list of all DoF rotation angles

index range: 0 ~ (DoF - 1)

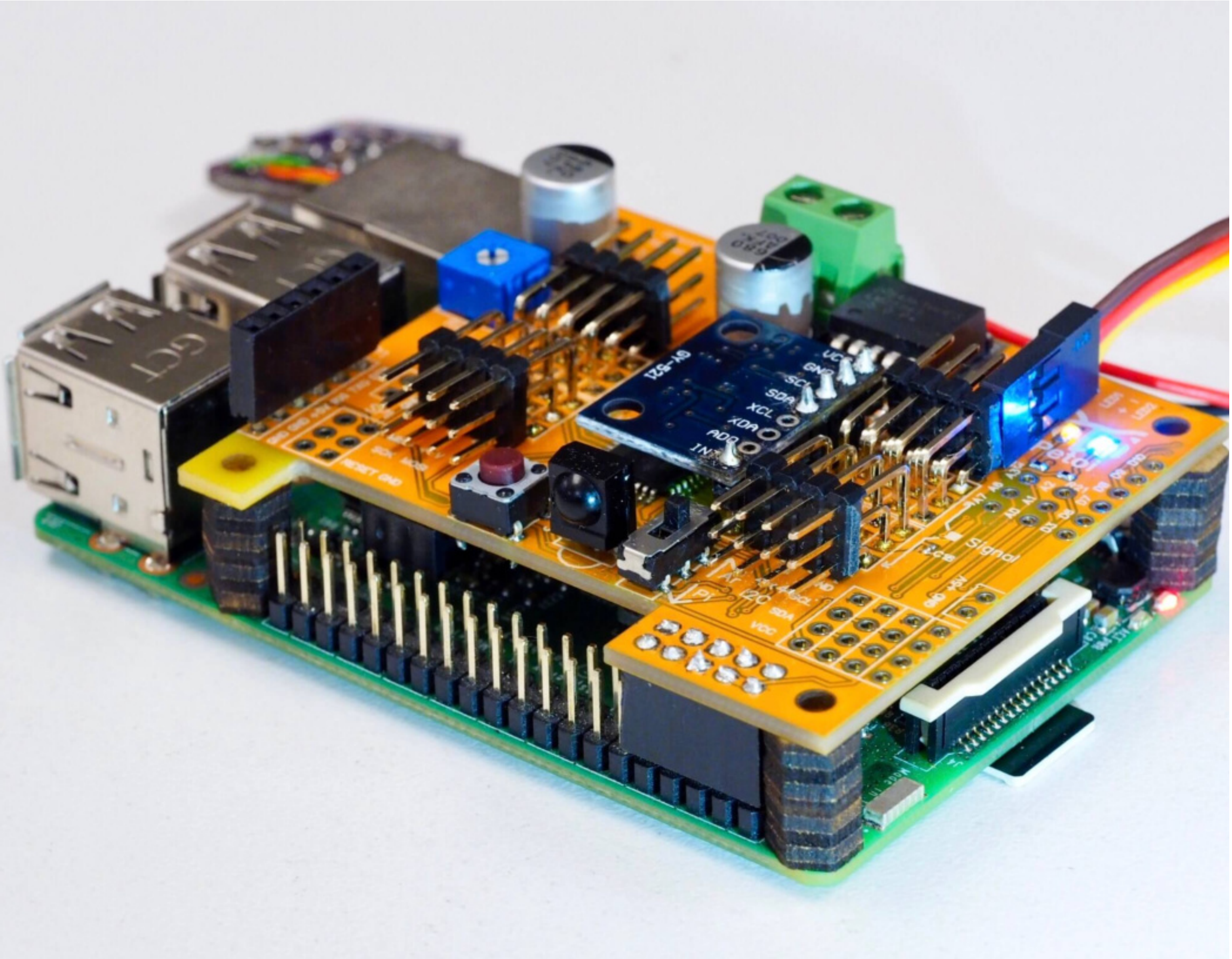
* angle range: -90 ~ 90. fits in the range of signed char (-128 ~ 127). Also depends on the servos' parameters

All the token starts with a single Ascii encoded character to specify its parsing format. They are case-sensitive and usually in lower case.

> ⓘ Some tokens haven't been implemented, such as 'h'. Token 'i' and 'l' still have some bugs.

4.4. Raspberry Pi serial port as an interface

> Only when using Pi as a master controller. Nybble doesn't need a Pi to move.
>
> You need to disconnect the serial uploader to communicate with Pi's serial port.



As shown in the serial protocol, the arguments of tokens supported by Arduino IDE's serial monitor are all encoded as Ascii char strings for human readability. While a master computer (e.g. RasPi) supports extra commands, mostly encoded as binary strings for efficient encoding. For example, when encoding angle 65 degrees:

- Ascii: takes 2 bytes to store Ascii characters '6' and '5'
- Binary: takes 1 byte to store value 65, corresponding to Ascii character 'A'

> What about value -113? It takes four bytes as an Ascii string but still takes only one byte in binary encoding, though the content will no longer be printable as a character.

Obviously, binary encoding is much more efficient than the Ascii string. However, the message transferred will not be directly human-readable. In the OpenCat repository, I have put a simple Python script ardSerial.py that can handle the serial communication between NyBoard and Pi.

4.4.1. Config Raspberry Pi serial port

In Pi's terminal, type `sudo raspi-config`

Under the **Interface** option, find **Serial**. Disabled the serial login shell and enable the serial interface.

> ⓘ A good tutorial on Instructable

If you plug Pi into NyBoard's 2x5 socket, their serial ports should be automatically connected at 3.3V. Otherwise, pay attention to the Rx and Tx pins on your own AI chip and its voltage rating. The Rx on your chip should connect to the Tx of NyBoard, and Tx should connect to Rx.

> ⚠ You also need to DISABLE the 1-wire interface of Pi to avoid repeating reset signals sent by Pi's GPIO 4.

4.4.2. Change the permission of ardSerial.py

If you want to run it as a bash command, you need to make it executable:

```
chmod +x ardSerial.py
```

You may need to change the proper path of your Python binary on the first line:

```
#!/user/bin/python
```

4.4.3. Use ardSerial.py as the commander of Nybble

> ⓘ NyBoard has only one serial port. You need to UNPLUG the FTDI converter if you want to control Nybble with Pi's serial port.

Typing `./ardSerial.py <args>` is almost equivalent to typing <args> in Arduino's serial monitor. For example, `./ardSerial.py kcr` means "perform s**k**ill **cr**awl".

Both **ardSerial.py** and the parsing section in **OpenCat.ino** need more implementations to support all the serial commands in the protocol.

> ⓘ Reduced motion capability may happen when connected to Pi! A stronger battery is needed.

With the additional current drawn by Pi, Nybble will be less capable for intense movements, such as trot (the token is `ktr`). The system is currently powered by two 14500 batteries in series. You may come up with better powering solutions, such as using high drain 7.4 Lipo batteries, or 2S-18650. There are a bunch of considerations to collaborate software and hardware for balanced performance. With Nybble's tiny body, it's better to serve as a platform for initiating the communication framework and behavior tree rather than a racing beast.

4.5. Battery

Though you can program NyBoard directly with the FTDI uploader, external power is required to drive the servos.

When powering the NyBoard with only USB FTDI, there's obviously charging and uncharging in the servo's capacitor and cause the yellow LED to pulse. However the USB's current is not sufficient to keep the servos working. The servo circuit has to be powered by external batteries to work properly.

4.5.1. Voltage

NyBoard requires 7.4~9V external power to drive the servos. On Nybble, we are using the 8V standard to configure all the parameters as a whole system. That's usually two Li-ion or Li-poly batteries connected in series. A single battery is 4.2V when fully charged and can work normally until voltage drops to 3.6V. That’s about 7.2V with two batteries connected in series. Before installation, dial the potentiometer on NyBoard clockwise to try minimum output first (about 4.5V), then dial it up until it can make the robot work properly. With our kit servos, it actually works better to connect SW3 between BATT with V_S directly and bypass the step-down chip (so no need to tune the potentiometer).

When looking for batteries, search for keywords “14500 3.7V li-ion battery unprotected”. I’ve noticed that the overcurrent protection of some batteries could be triggered by peak current draw-(usually >2.5A), causing NyBoard to reset or malfunction. Try to find batteries with higher discharge ratings.

Read this forum post to find batteries that work on Nybble.

4.5.2. Dimensions

The included battery holder is sized for 14500 batteries, which is 14 mm in diameter, and 50 mm in length. 50 ± 1 mm should still fit in. They are the same size as AA batteries, but much more powerful. **Make sure not to use them in regular AA devices.** If you are in the US, we have tested with EBL 14500 li-ion batteries.

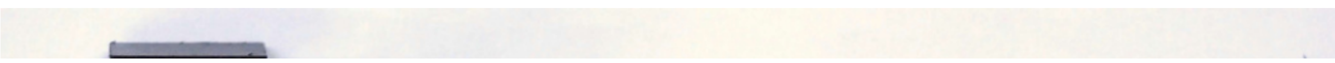
You can also design other battery holders to carry larger batteries for better performance. That’s especially necessary if you mount a Raspberry Pi or want Nybble to run faster.

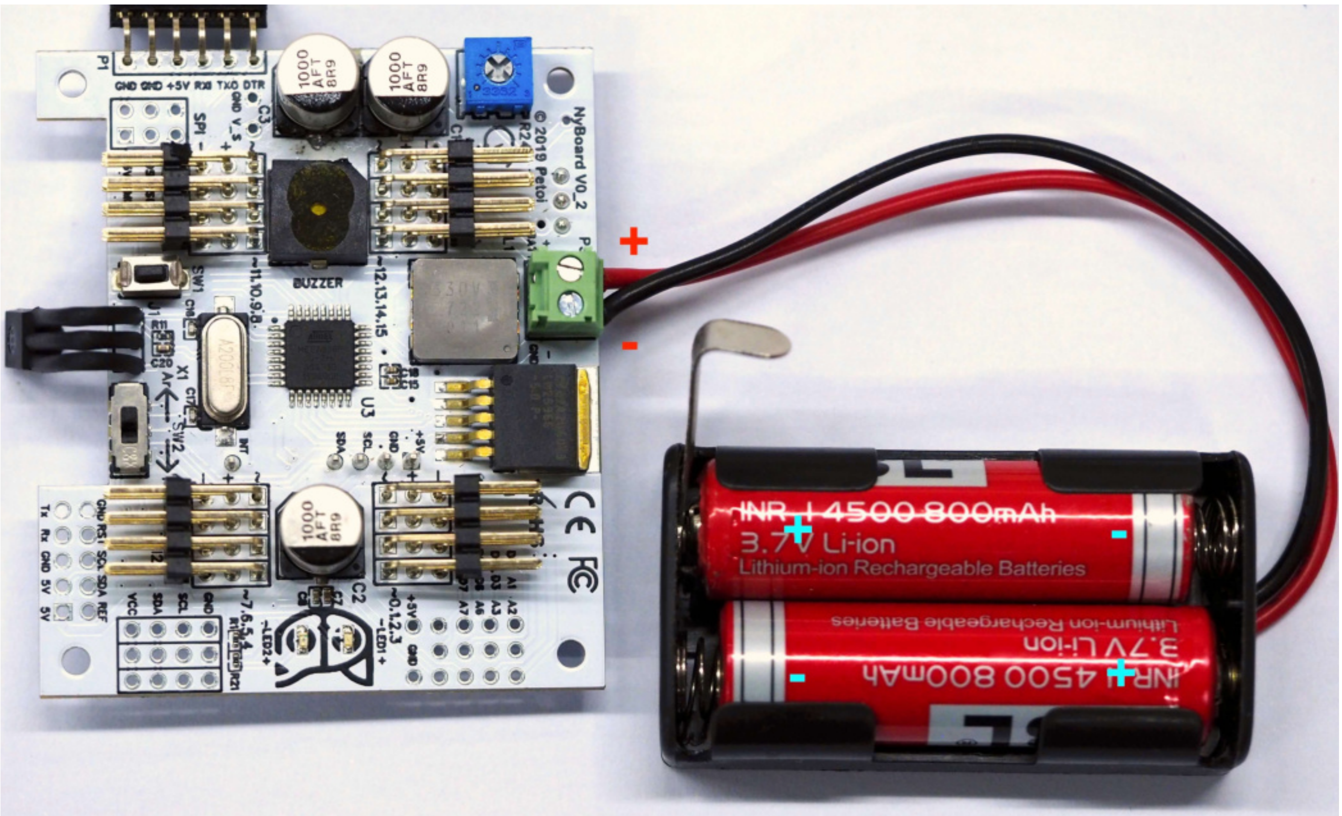
4.5.3. Connection

Be careful with the polarity when connecting the power supply. Make sure you can find the positive (+) and negative (-) sign on both the NyBoard's power terminal and your power supply.

> ⚠ **Reversed connection may damage your NyBoard!**

Loosen the screws of the power block. Insert the wires of the battery holder then tighten the screws. When turning the switch on, both the blue LED (for the chip) and the yellow LED (for servo) should lit up.





In the kit shipped after Jan.1st, the terminal is replaced with an anti-reverse socket, so you won't be able to plug in the wrong direction.

4.5.4. Battery life varies according to usage

It can last hours if you're mainly coding and testing postures, or less than 30 mins if you keep Nybble running.

When the battery is low, the yellow LED will blink slowly. Although NyBoard can still drive one or two servos, it will be very unstable to drive multiple servos at once. **That will lead to repeatedly restarting the program or awkward joint rotations.** In rare cases, it may even alter the bits in EEPROM. You will need to reupload the codes and re-save the constants to recover.

4.5.5. Charging

You will need compatible smart chargers for the batteries. Keep batteries attended during charging.

4.5.6. After use

After playing, remember to **remove the batteries from the battery holder to avoid over-discharging**.

4.5.7. Signal interference

It's ok to connect both FTDI and battery at the same time. You can type in serial commands while the battery is connected. I do notice that the USB serial port could be disabled randomly. I think that's due to the sudden current draw by servos. It will trigger the computer’s overcurrent protection and disable the USB port. In that

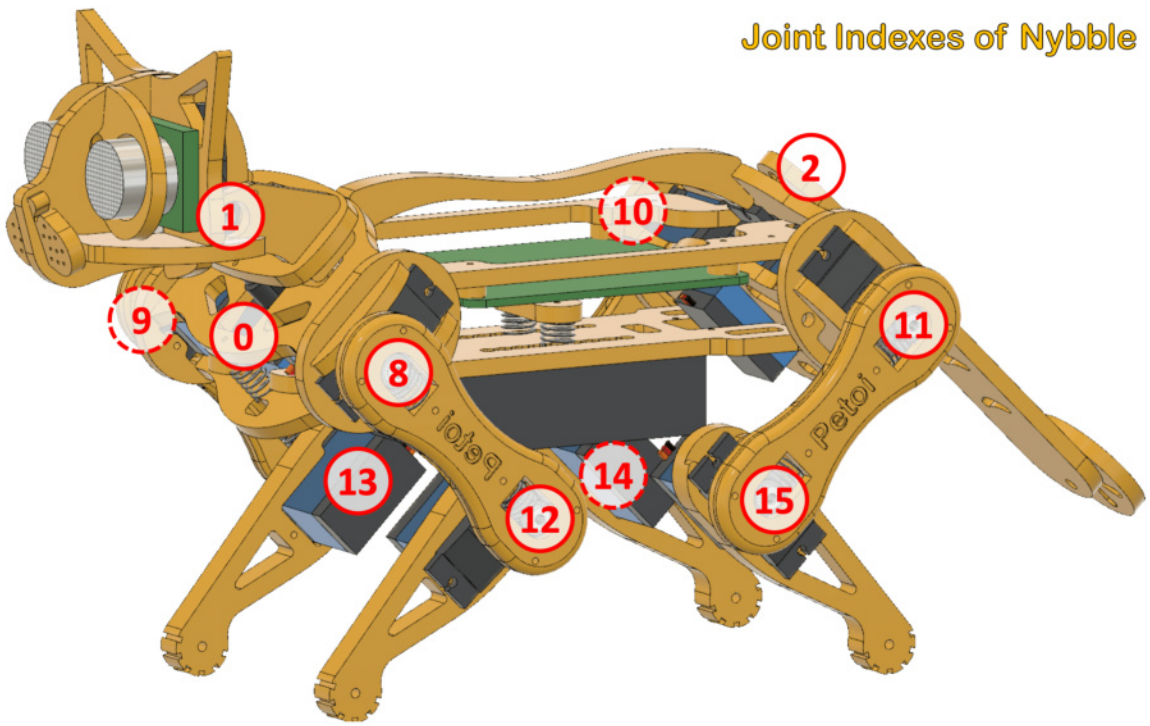
case, you can change the USB port you're connecting to, reset the USB bus, or restart the computer. So

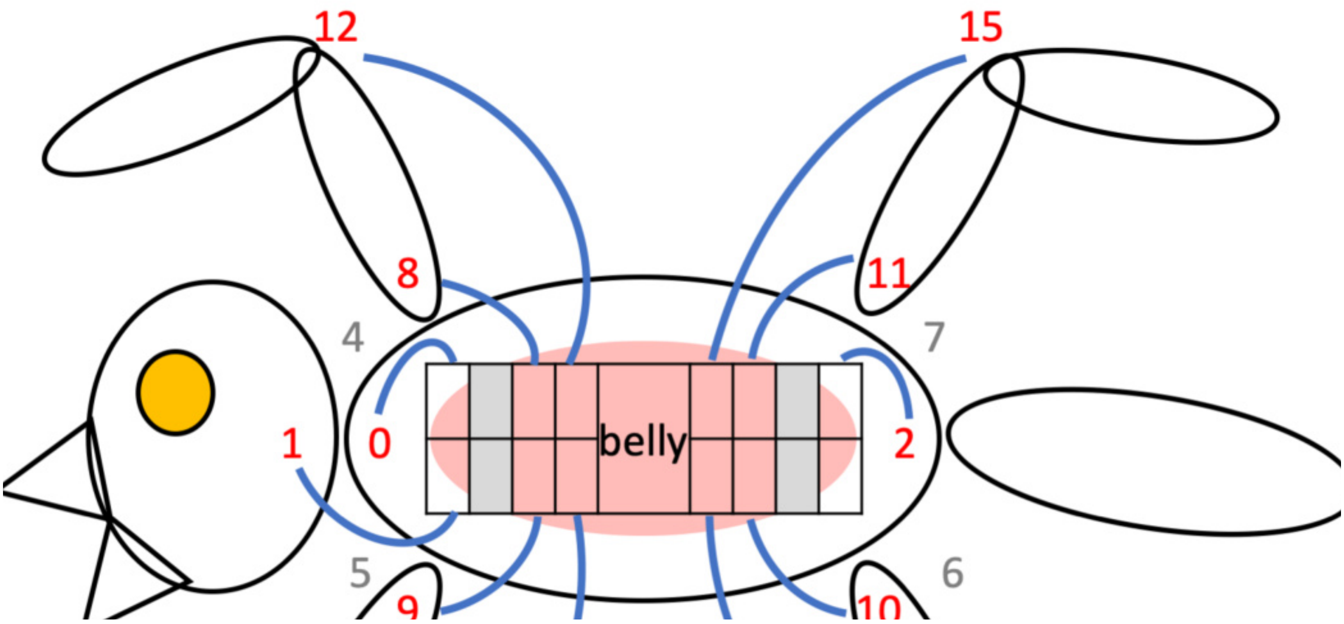
5 Connect Wires

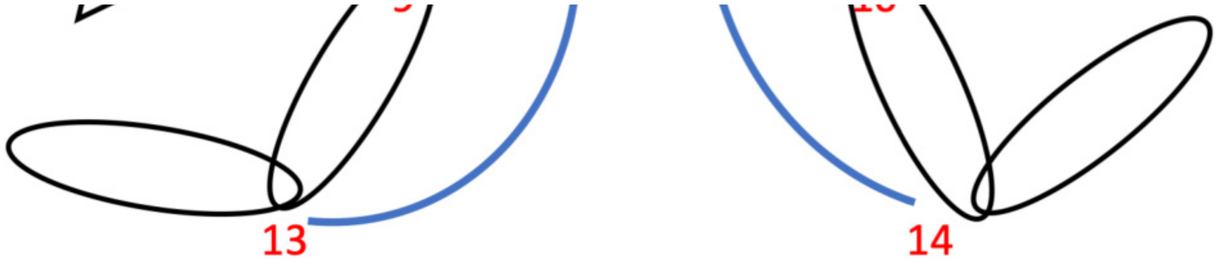
"Everything is connected."

5.1. Joint map

Nybble's servos are connected to NyBoard's PWM pins symmetrically and resemble the nerves along the spinal cord. Though Nybble doesn't have shoulder roll DoF, those indexes(4~7) are reserved for the full OpenCat framework.

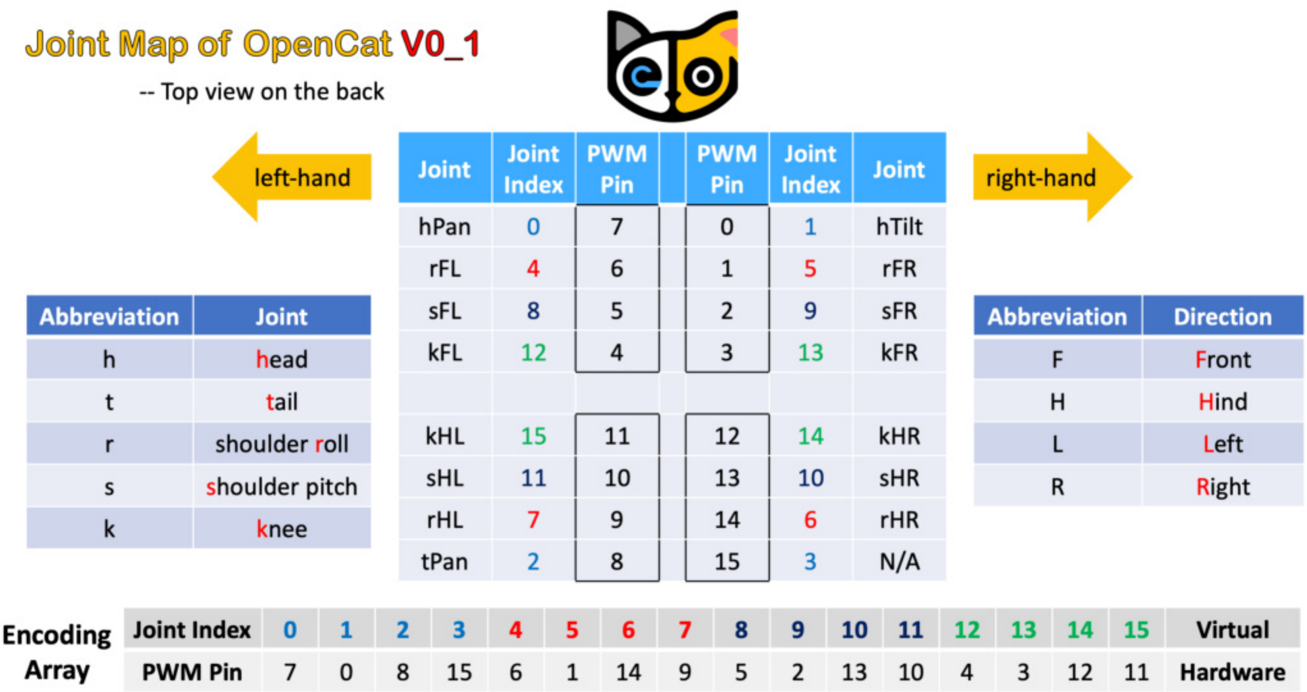






> ⓘ NyBoard V0_* and V1_* have different PWM pin orders, but the physical servo connection remains the same. All adjustments are handled in the software. **You don't even need to read the pin numbers on the PCB board.**

Use h for the head, t for the tail, r for shoulder roll joint, s for shoulder pitch joint, k for knee joint, F for the front, H for hind, L for left, R for right, the full joints map of OpenCat is:



Joint	Joint Index	PWM Pin		PWM Pin	Joint Index	Joint
hPan	0	7		0	1	hTilt
rFL	4	6		1	5	rFR
sFL	8	5		2	9	sFR
kFL	12	4		3	13	kFR
kHL	15	11		12	14	kHR
sHL	11	10		13	10	sHR
rHL	7	9		14	6	rHR
tPan	2	8		15	3	N/A

Abbreviation	Joint
h	head
t	tail
r	shoulder roll
s	shoulder pitch
k	knee

Abbreviation	Direction
F	Front
H	Hind
L	Left
R	Right

Encoding Array

Joint Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Virtual
PWM Pin	7	0	8	15	6	1	14	9	5	2	13	10	4	3	12	11	Hardware



Joint	Joint Index	PWM Pin		PWM Pin	Joint Index	Joint
hPan	0	4		3	1	hTilt
rFL	4	5		2	5	rFR
sFL	8	6		1	9	sFR
kFL	12	7		0	13	kFR
kHL	15	8		15	14	kHR
sHL	11	9		14	10	sHR

Abbreviation	Joint
h	head
t	tail
r	shoulder roll
s	shoulder pitch

Abbreviation	Direction
F	Front
H	Hind
L	Left
R	Right

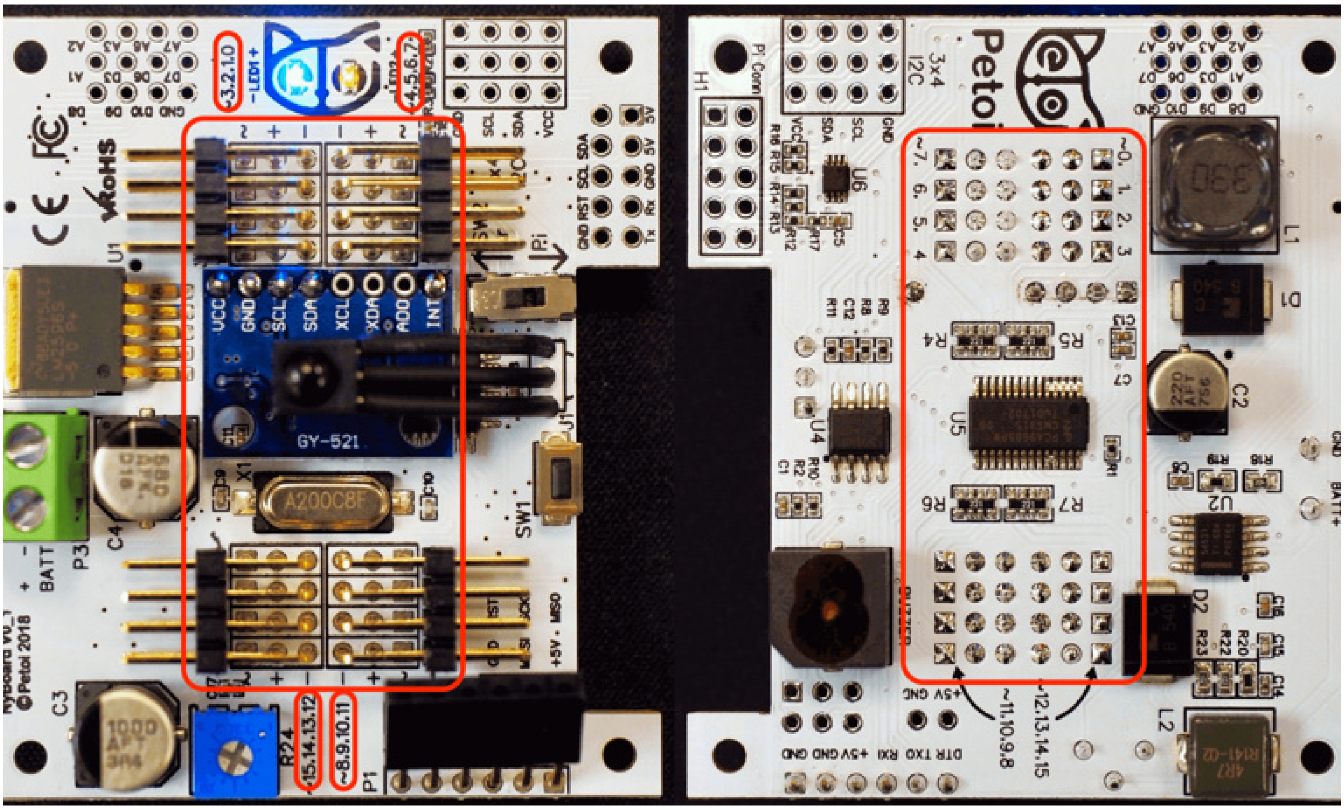
k	knee

rHL	7	10	13	6	rHR
tPan	2	11	12	3	N/A

Encoding Array	Joint Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Virtual
	PWM Pin	4	3	11	12	5	2	13	10	6	1	14	9	7	0	15	8	Hardware

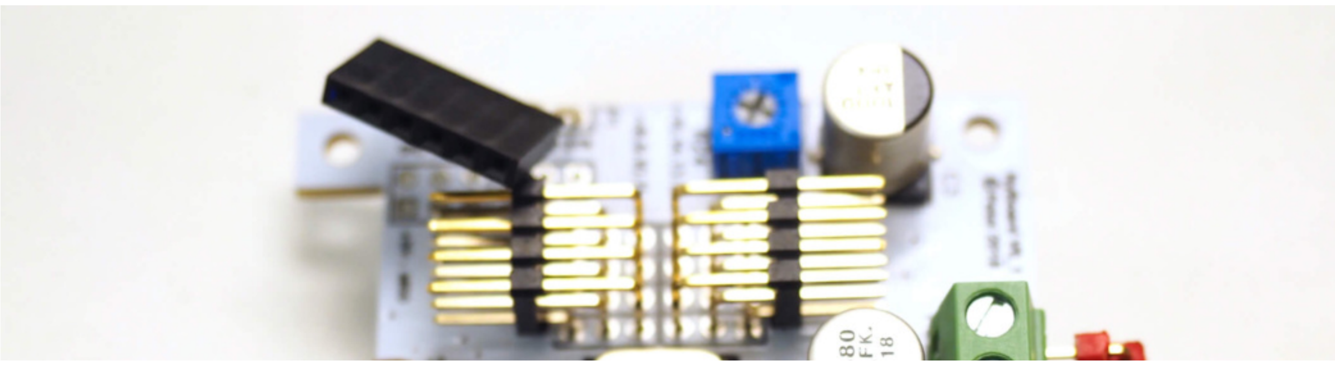
5.2. Plug in the servos

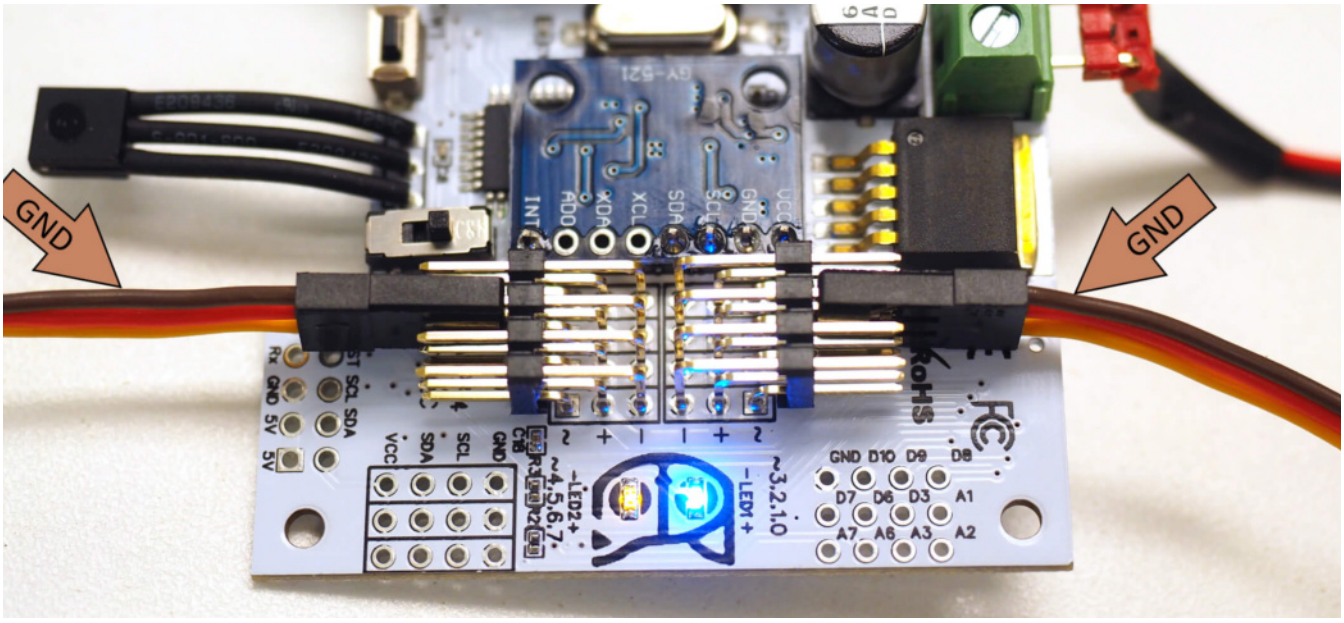
Observe the indexing pattern to connect servos with correct PWM pins. Be careful with the wires' direction. The brown wire of the servo is GND, while the GND on NyBoard V0_1 is along the centerline. In later versions of NyBoards, they are opposite.



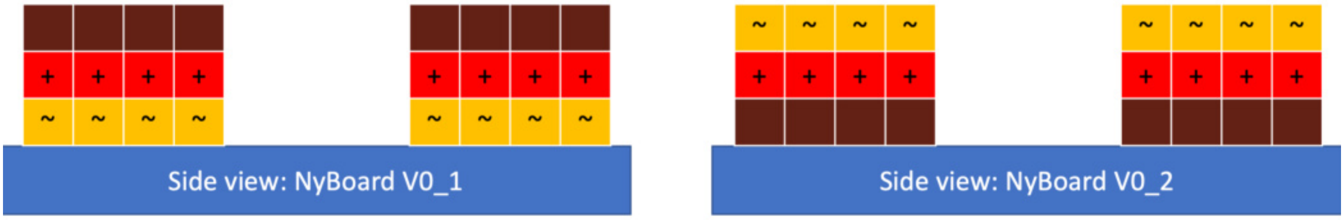
(NyBoard V0_1 Front) (NyBoard V0_1 Back)

A quick check is that all On NyBoard V0_1 the brown wires should be far away from the board surface. On NyBoard V0_2 the brown wires should be close to the board surface.

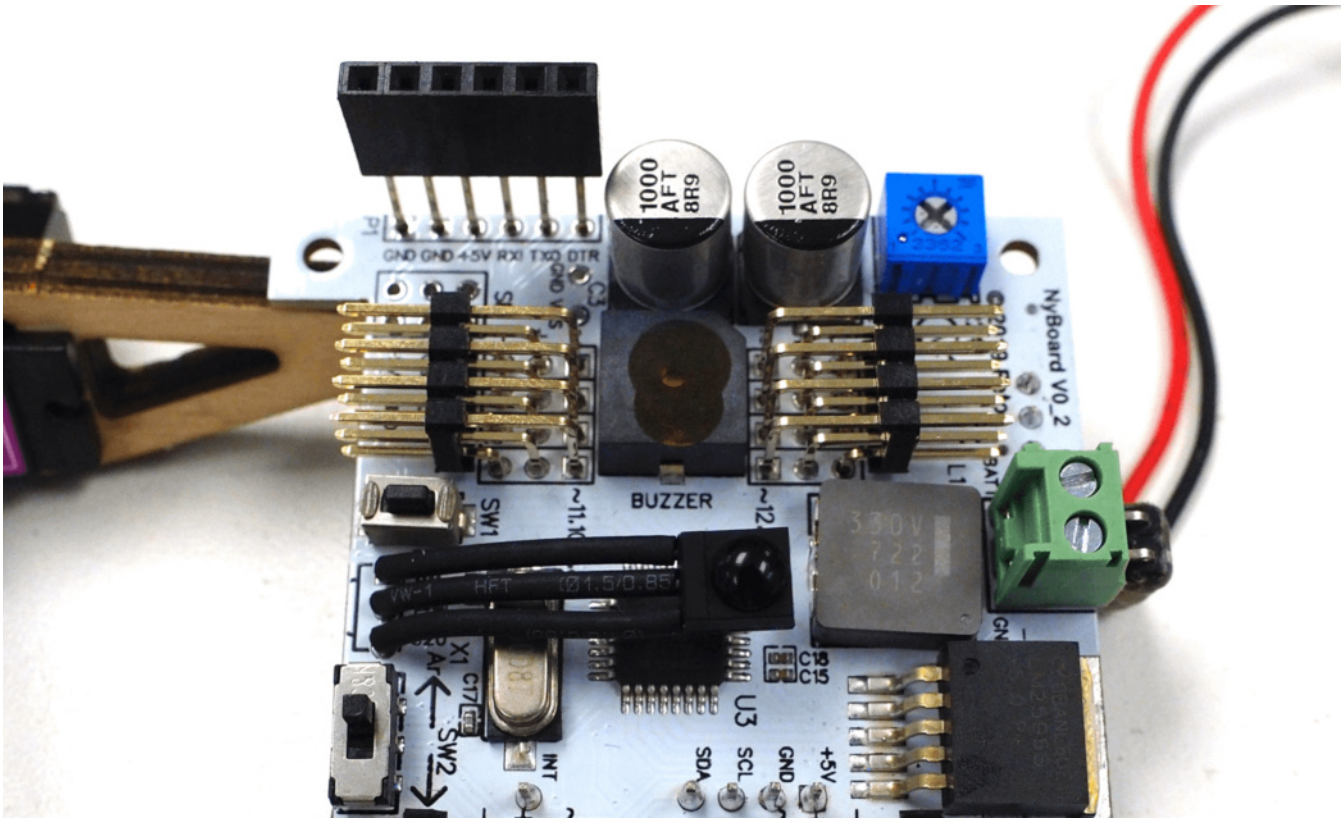


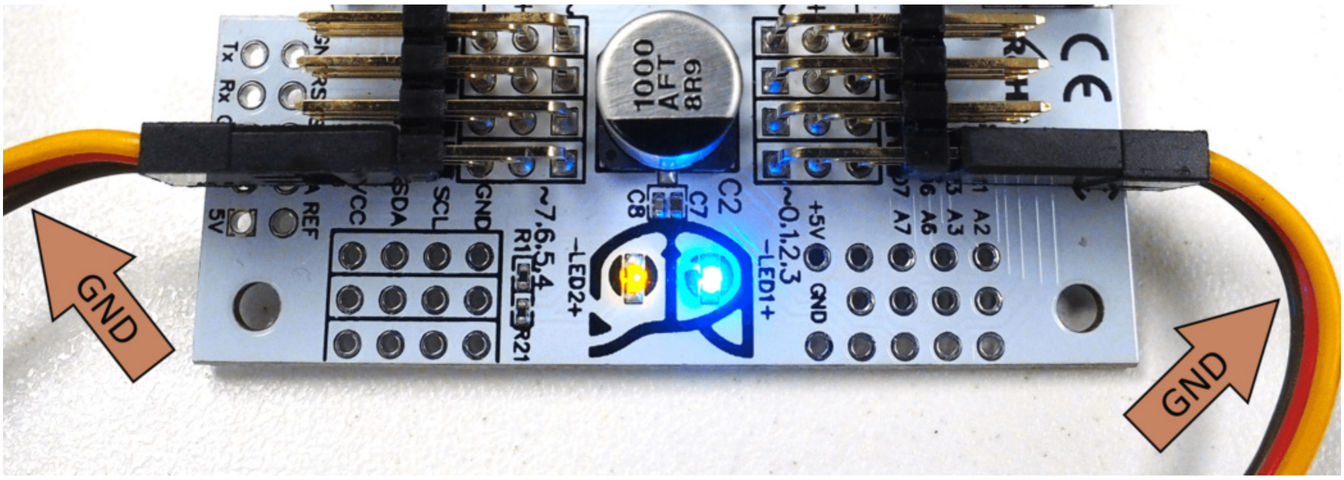


NyBoard V0_1



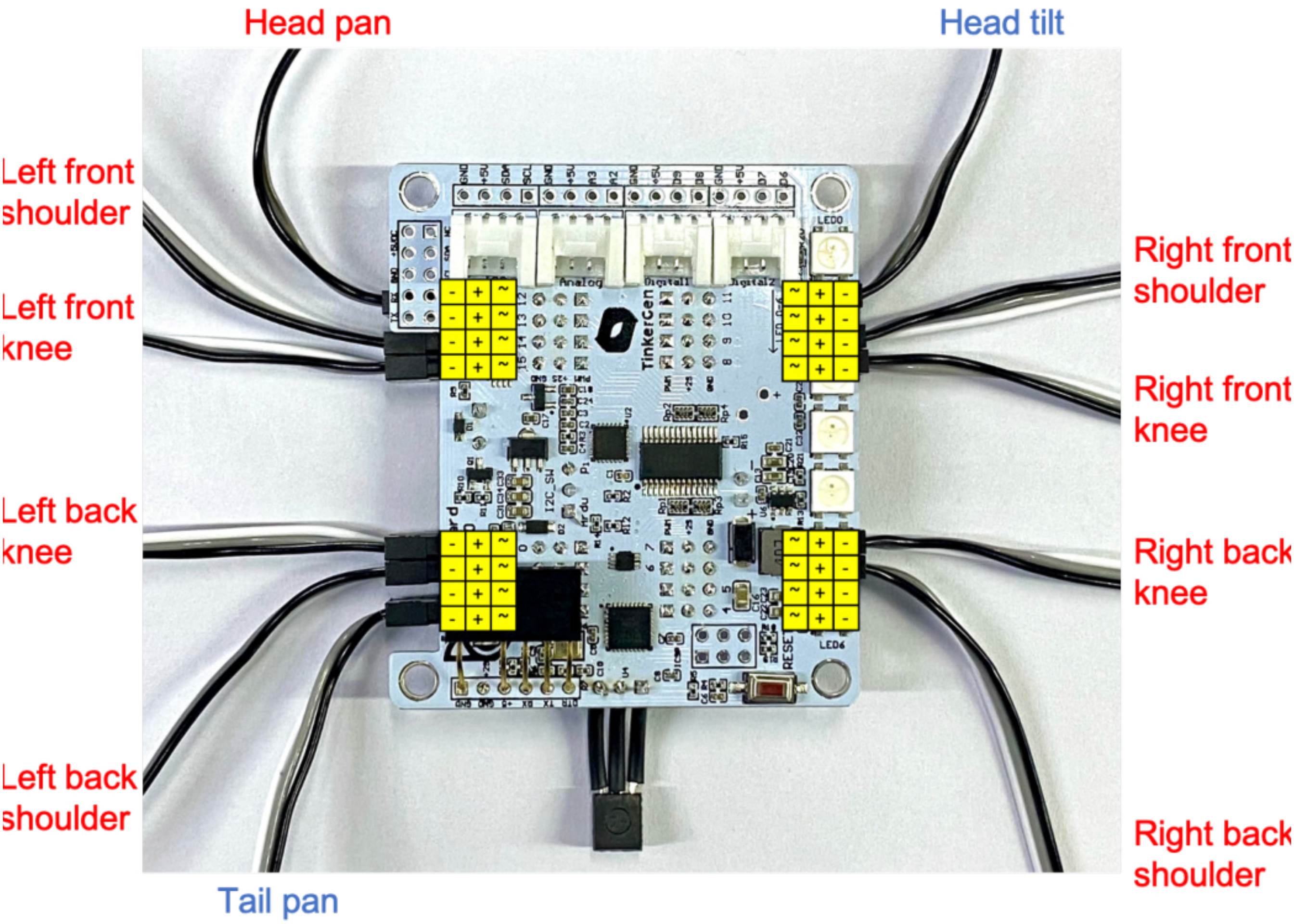
> ⓘ From NyBoard V0_2, the location of servo wires will still be the same, but their direction of colors will be opposite





NyBoard V0_2

Even though the PCB layouts vary on different NyBoards, the wiring pattern is the same. You don't need to read the indexes of the servo pins when plugging the servos. The joint mapping is done automatically in the OpenCat code. You just need to select the right board version when uploading the code.



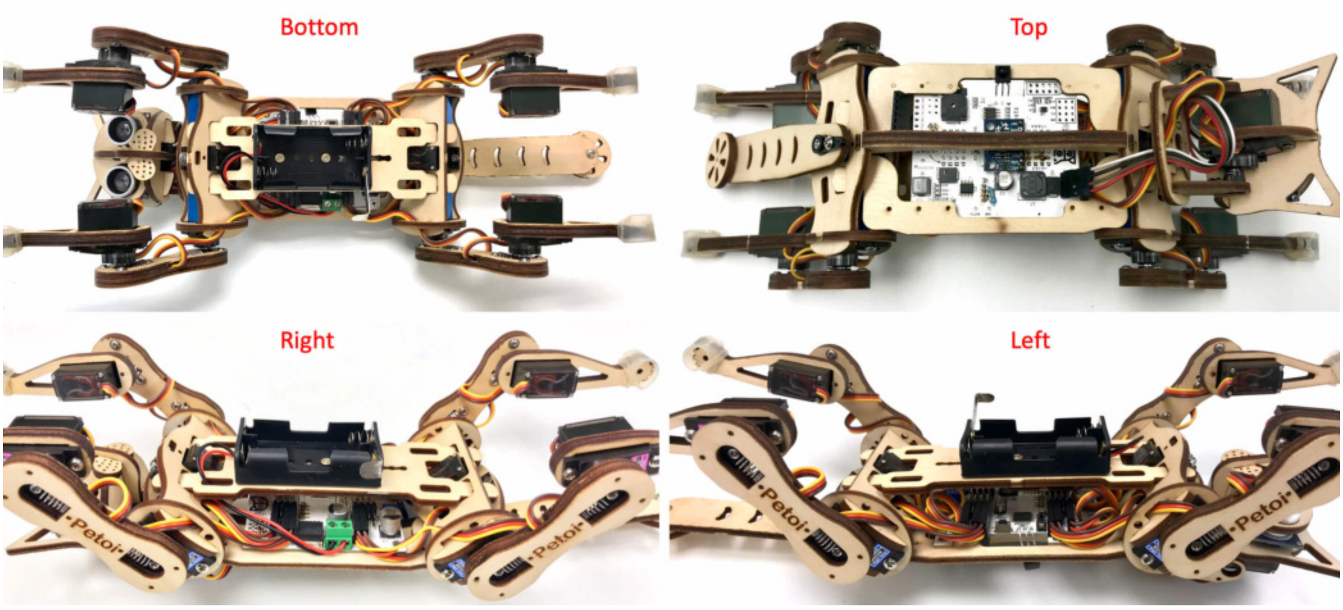
Wiring pattern of NyBoard V1_0

5.3. Wiring

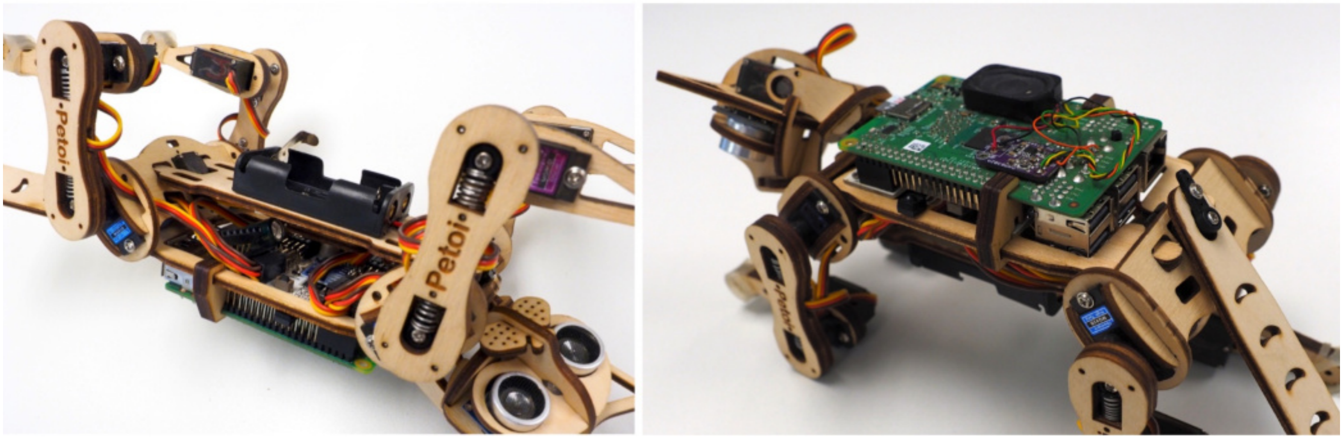
After calibration, troubleshooting and final assembling, It’s time to think about wiring routes to make Nybble

look neat. There are multiple slots on the frame designed as a wire organizer. Feel free to develop your own style for connecting future accessories. Make sure the wires don't get in the way of the servos' movement.

Below is my wiring routes.



NyBoard alone



NyBoard with Raspberry Pi

5.4. Connect Ultrasonic Sensor (optional)

If you have not already, plug in the 4 pin cable, make a note of which cable colors are attached to which signals. When looking into Nybble's eyes, the connections from left to right are:

VCC	Trigger	Echo	Ground

Connect the other end of the cable to the header you soldered in place earlier in assembly. Use the following mapping

Sensor Side	NyBoard Side

Vcc	D8
Trigger	D9
Echo	D10
GND	GND

6 Calibration

"A miss is as good as a mile."

Calibration is vital for Nybble to work properly.

In previous sections, we have prepared those body parts, but haven't screwed them onto servos. If we don't calibrate the servos before attaching them, they may rotate to any direction, get stuck, and cause damage to either the servos or body parts.

The calibration has four steps:

1. Write constants to the board
2. Power on the circuit, let servos rotate freely to zero angle/calibration state
3. Attach body parts to the servos
4. Fine-tune the offsets in software.

The logic behind calibration can be found on the OpenCat forum.

6.1. Write constants

6.1.1. There are three types of constants to be saved to NyBoard:

1. Assembly related definitions, like joint mapping, rotation direction, sensor pins. They are pretty fixed and are mostly defined in **OpenCat.h**. They are even kept consistent with my future robots;
2. Calibration related parameters, like MPU6050 offsets and joint corrections. They are measured in realtime and are saved in the onboard EEPROM. They only need to be measured once;
3. Skill related data, like postures, gaits, and pre-programmed behaviors. They are mostly defined in **Instinct.h**. You can add more customized skills too.

6.1.2. Upload and run WriteInstinct.ino.

The role for **WriteInstinct.ino** is to write constants to either onboard or I2C EEPROM, and save calibration

> ⓘ You need to change the ***** on `#define NyBoard_V*_*` in **Instinct.h** to match your NyBoard's version. The version number is to the left of the green battery terminal.

> ⓘ Before uploading the sketch, make sure to close all opened windows of the serial monitor. Otherwise, the serial port may be occupied and the sketch will not be uploaded.

> ⓘ You also need to dial the slide switch on NyBoard to **Arduino** rather than Pi!

After finish uploading **WriteInstinct.ino**, open the serial monitor.

> ⓘ Make sure to set the serial monitor as **115200 baud rate and no line ending**.



You will see several questions:

`Reset all joint calibration? (Y/n)`

If you have never calibrated the joints, or if you want to recalibrate the servos with a fresh start, type 'Y' to the question. The 'Y' is CASE SENSITIVE!

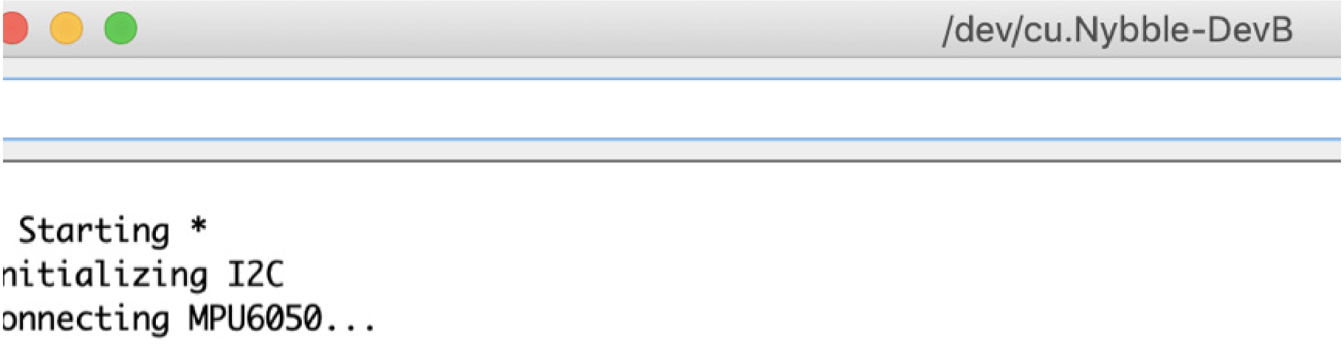
`Do you need to update Instincts? (Y/n)`

If you have modified the Instinct.h in any way, you should type 'Y'. Though it's not always necessary once you have a deeper understanding of memory management.

`Calibrate MPU? (Y/n)`

If you have never calibrated the MPU6050, i.e. the gyro/accelerometer sensor, type 'Y'.

Sometimes the program could hang at the connection stage. You can close the serial monitor and reopen it, or press the reset button on NyBoard, to restart the program.



6.2. Enter calibration mode

The calibration state is defined as the middle point of the servo's reachable range. Calibration for servos can be done in either **WriteInstinct.ino** or **OpenCat.ino**. I recommend you do it with **WriteInstinct.ino** in case there's something wrong with the constants.

You MUST plug in all the servos and batteries for proper calibration. Then in the serial monitor, type 'c' to enter calibration mode. The servos should rotate one by one with unnoticeable time intervals then stop. Depending on their initial shaft direction, some may travel larger angles until stopping at the middle point. There will be noise coming from the gear system of the servos. You will see the calibration table:

```
 1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11,  12,  13,  14,  15
-1  -1  -1  -1  -1  -1  -1  -1  -1  -1   -1   -1   -1   -1   -1
 0
```

The first row is the joint indexes, the second row is their calibration offsets:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Offset	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	

Initial values are "-1" or "0", and should be changed by later calibration.

> ⓘ The servos are using a potentiometer in the feedback loop for position control. When holding at a static position, they tend to vibrate around the target angle. A Parkinson's-like vibration will develop after a short period of use. It won't affect much during continuous motion. Better servos

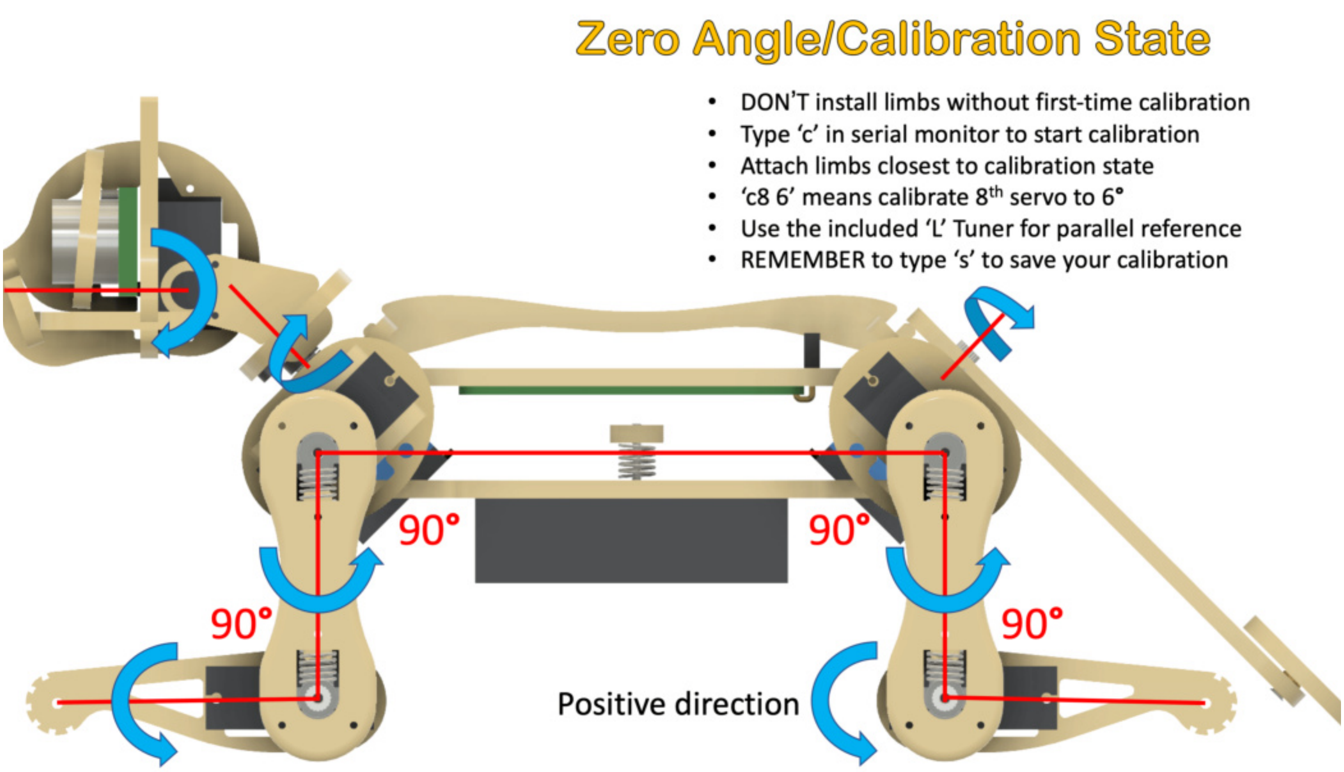
> without these troubles could cost 10 times more, so replacing a failed unit is a more cost-effective solution.

6.3. Attach head, tail, and legs.

6.3.1. Coordinate system

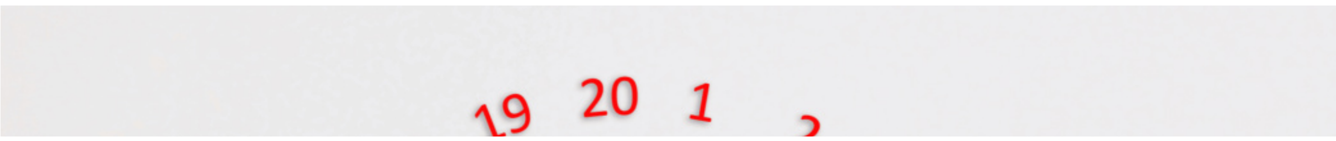
After entering ‘c’, with all servos rotated to their zero angles, now attached the head, tail, and legs prepared in the previous section to the body. They are generally perpendicular to their linked body frames. Avoid rotating the servo shaft during the operation.

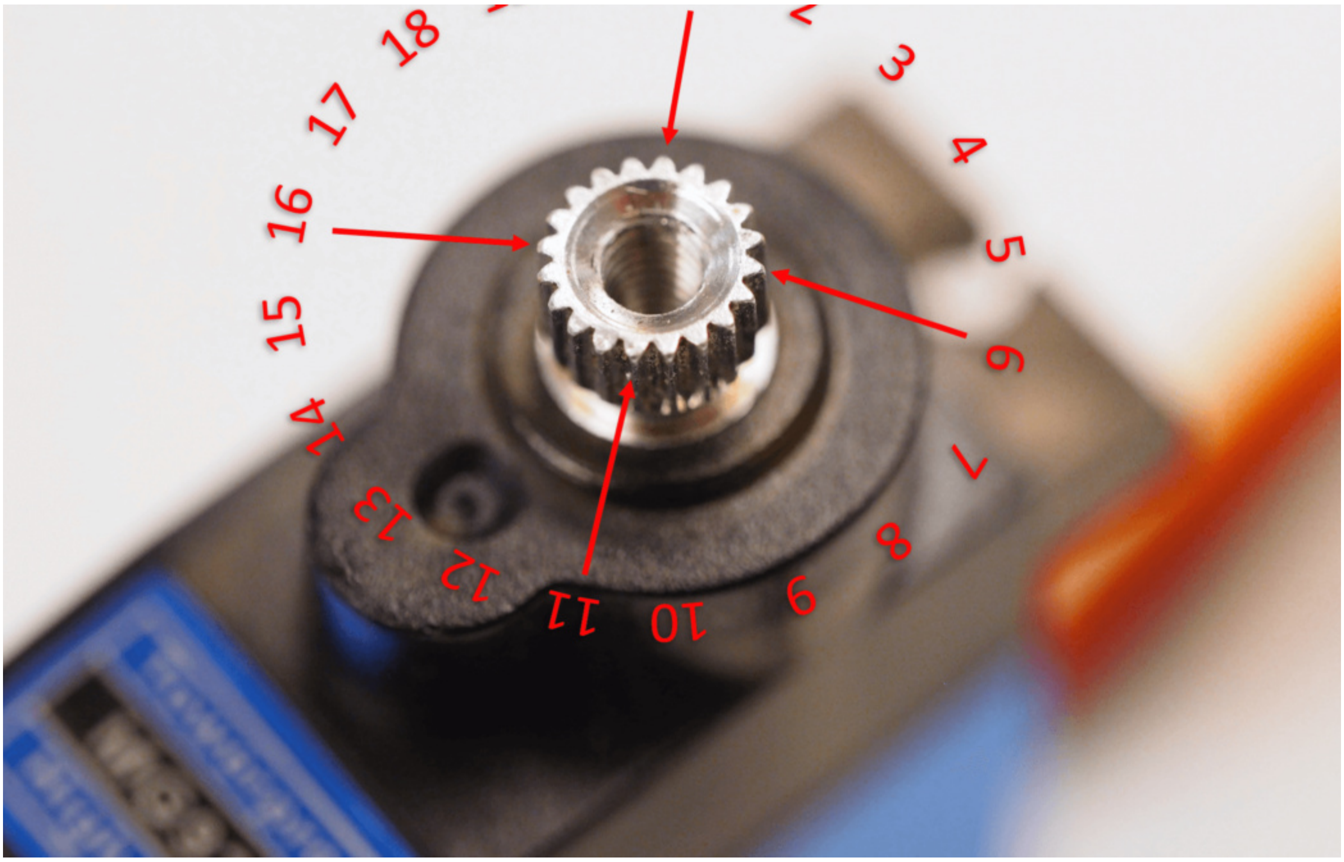
Rotating the limbs counter-clockwise from their zero states will be positive (same as in polar coordinates). The only exception is the tilt angle for the head. It’s more natural to say head up, while it’s the result of rotating clockwise.



6.3.2. Understand the angle divisions

If we take a closer look at the servo shaft, we can see it has a certain number of teeth. That’s for attaching the servo arms, and to avoid sliding in the rotational direction. In our servo sample, the gears are dividing 360 degrees to 20 sectors, each taking **18** degrees. That means we cannot always get exact perpendicular installation. But try to get them as close as possible to their zero states. Use screw A to fix the limbs onto servos.





6.4. Find and save calibration offsets

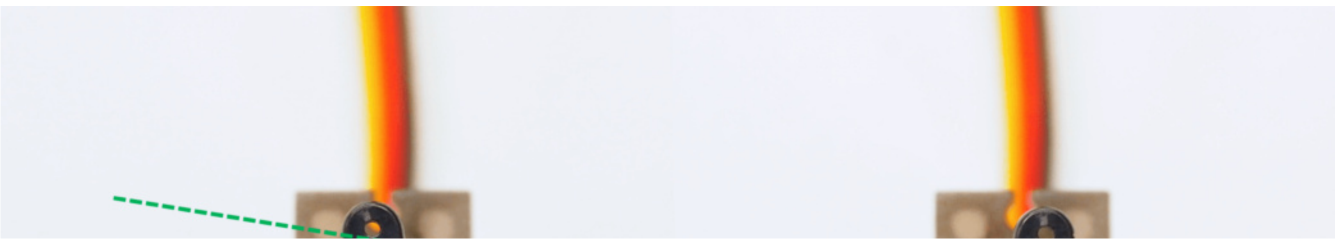
6.4.1. Fine-tune the calibration on the software side

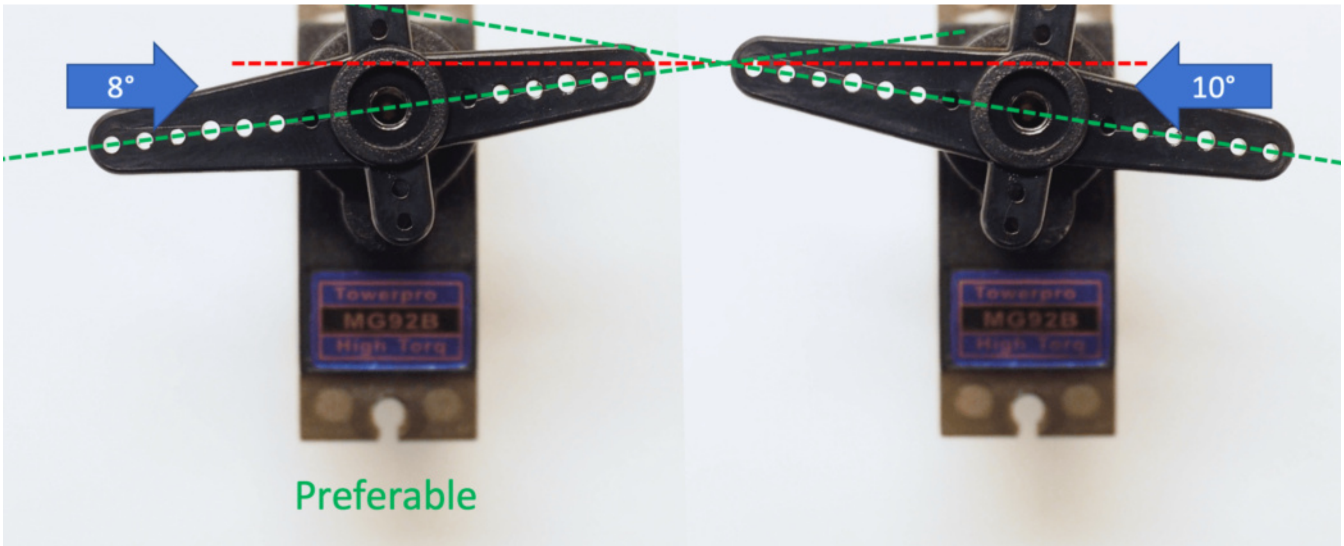
The command for calibration (refer to the serial communication protocol for NyBoard) is formatted as `cIndex Offset`. Notice that there's a space between Index and Offset.

For example, `c8 6` means giving the 8th servo an offset of 6 degrees. Find the best offset that can bring the limb to the zero state.

> ⓘ If you find the absolute value of offset is larger than 9, that means you are not attaching the limb closest to its zero state. That will result in a decreased reachable range of the servo on either side. Take off the limb and rotate it by one tooth. It will result in an opposite but smaller offset.

For example, if you have to use -13 as the calibration value, take the limb off, rotate by one tooth then attach back. The new calibration value should be around 5, i.e., they sum up to 18. Avoid rotating the servo shaft during this adjustment.



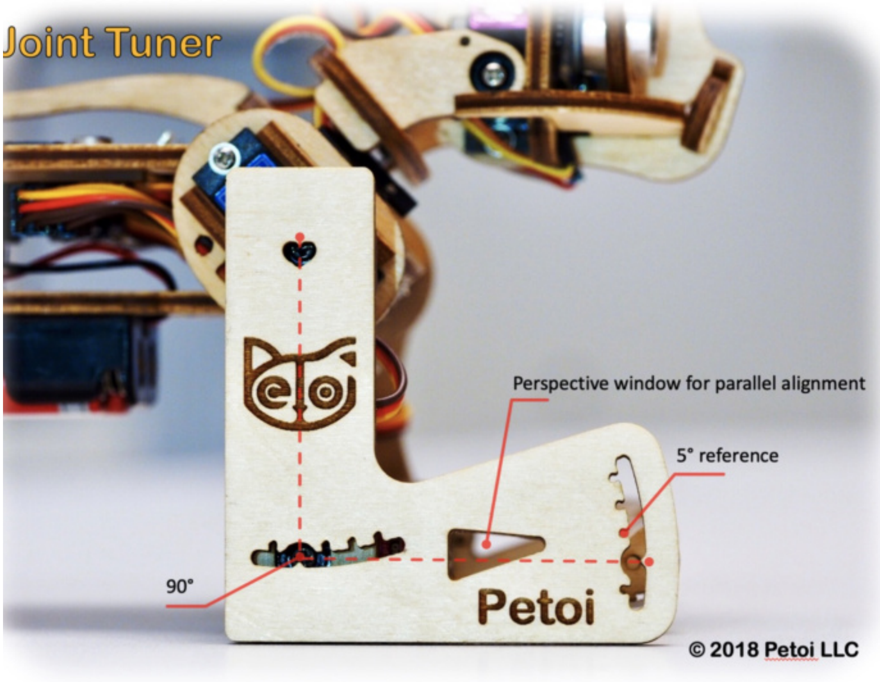


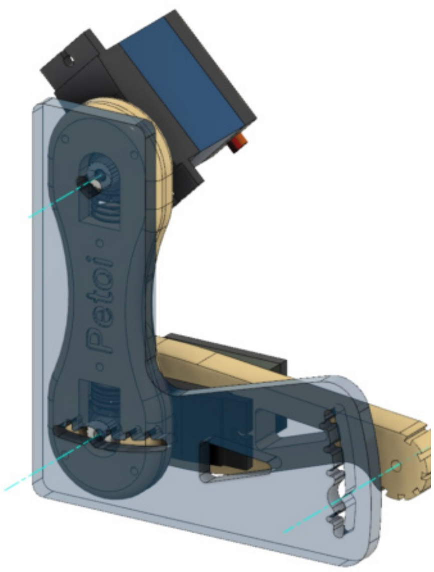
After calibration, remember to type ‘s’ to save the offsets. Otherwise, they will be forgotten when exiting the calibration state. You can even save every time after you’re done with one servo.

6.4.2. ‘L’ shaped joint tuner

When watching something, the observation will change from different perspectives. That’s why when measuring length, we always want to read directly above the ruler.

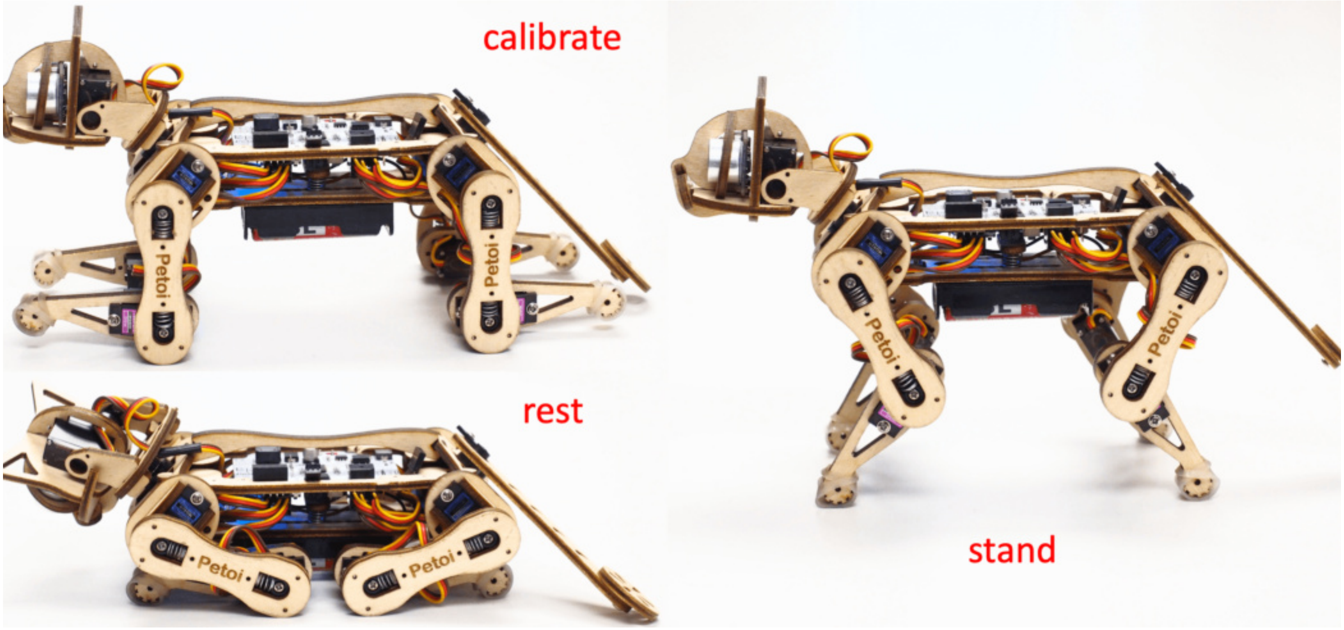
It’s especially important that you keep a parallel perspective when calibrating Nybble. Use the 'L' shaped joint tuner as a parallel reference to avoid reading errors. Align the tips on the tuner with the center of the screws in the shoulder and knee joints, and the little hole on the tip of the foot. Look along the co-axis of the centers. For each leg, calibrate shoulder servos (indexed 8~11) first, then the knee servos(indexed 12~15). When calibrating the knee, use the matching triangle windows on both the tuner and shank to ensure parallel alignment.





6.4.3. Validation

After calibration, type ‘d’ or ‘kbalance’ to validate the calibration. It will result in Nybble symmetrically moving its limbs between rest and stand state.



6.4.4. Center of mass

Try to understand how Nybble keeps balance even during walking. If you are adding new components to Nybble, try your best to distribute its weight symmetrically about the spine. You may also need to slide the battery holder back and forth to find the best spot for balancing.

7 Play with Nybble

"You can't direct the wind, but you can adjust your sails."

7.1. Control with Arduino IDE

Try the following serial commands in the serial monitor:

- “**k**sit”
- “**m**0 30”
- “**m**0 -30”
- “**k**balance”
- “**k**tr”
- “**k**trL”
- “**d**”

> ⓘ The quotation mark just indicates that they are character strings. Don’t type quotation marks in the serial monitor.

7.2. Control with Infrared remote

7.2.1. Keymap

Only the position of the buttons matters, though those symbols can help you remember the functionalities. I'm going to define position related symbols to refer to those keys.

I'm using abbreviations for key definitions to reduce SRAM usage. Due to the limited keys of a physical remote, I always change the definitions for fun.

> ⓘ The following map is just an illustration. Check function **String translateIR(){...}** in **OpenCat.ino** for the actual key definitions in effect. They are also open to your customization.

rest	F	gyro
L	balance	R
shutdown servos	B	calibrate
stepping	crawl	walk
trot	sit	stretch
greeting	push up	pee
look up	butt up	zero

- **Rest** puts the robot down and shuts down the servos
- Pressing **F/L/R** will make the robot to move forward/left/right
- **B** will make the robot move backward
- **Calibrate** puts the robot into calibration posture and turns off the gyro
- **Stepping** lets the robot step at the original spot
- **Crawl/walk/trot** are gaits that can be switched and combined with the direction buttons
- Buttons after **trot** are preset postures or other skills
- **Gyro** will turn on/off the gyro for self-balancing. Turning off the gyro can accelerate and stabilize the slower gaits. But it's NOT recommended for faster gaits such as trot.

We also made a customized remote panel for future batches. Previous users can download the design file and print it on A4 paper.

newPanel.pdf 2MB
PDF





7.2.2. Check out the following featured motions

- **Rest** puts the robot down and shuts down the servos. It's always safe to click it if Nybble is doing something **AWKWARD**. I'm serious. There are still some ghosts in the system I don't fully understand.
- **Balance** is the neutral standing posture. You can push Nybble from the side, or make it stand up will hind legs and tail. You can test its balancing ability on a fluctuating board. Actually balancing is activated in most postures and gaits.
- Pressing **F/L/R** will make the robot move forward/left/right
-

B will make the robot move backward

- **Calibrate** puts the robot into calibration posture and turns off the gyro
- **Stepping** lets the robot step at the original spot
- **Crawl/walk/trot** are the gaits that can be switched and combined with the direction buttons
- Buttons after **trot** are preset postures or other skills
- **Gyro** will turn on/off the gyro for self-balancing. Turning off the gyro can accelerate and stabilize the slower gaits. But it's NOT recommended for faster gaits such as trot.
- Lift Nybble at the middle of its spine so that all its legs can move freely in the air. Click all the buttons on the IR remote to see what they do. Then put Nybble on a wide flat table and try those buttons again. Different surfaces have different friction and will affect walking performance. The carpet will be too bushy for Nybble's short legs. It can only crawl (command **kcr**) over this kind of tough terrain.
- You can pull the battery pack down and slide along the longer direction of the belly. That will tune the center of mass, which is very important for walking performance. Otherwise, it may keep falling down.
- When Nybble is walking, you can let it climb up/down a small slope (<10 degrees)
- Whatever Nybble is doing, you can lift it vertically, and it will stop moving, just like a cat scruffed on the neck.

Buzzer beep meaning:

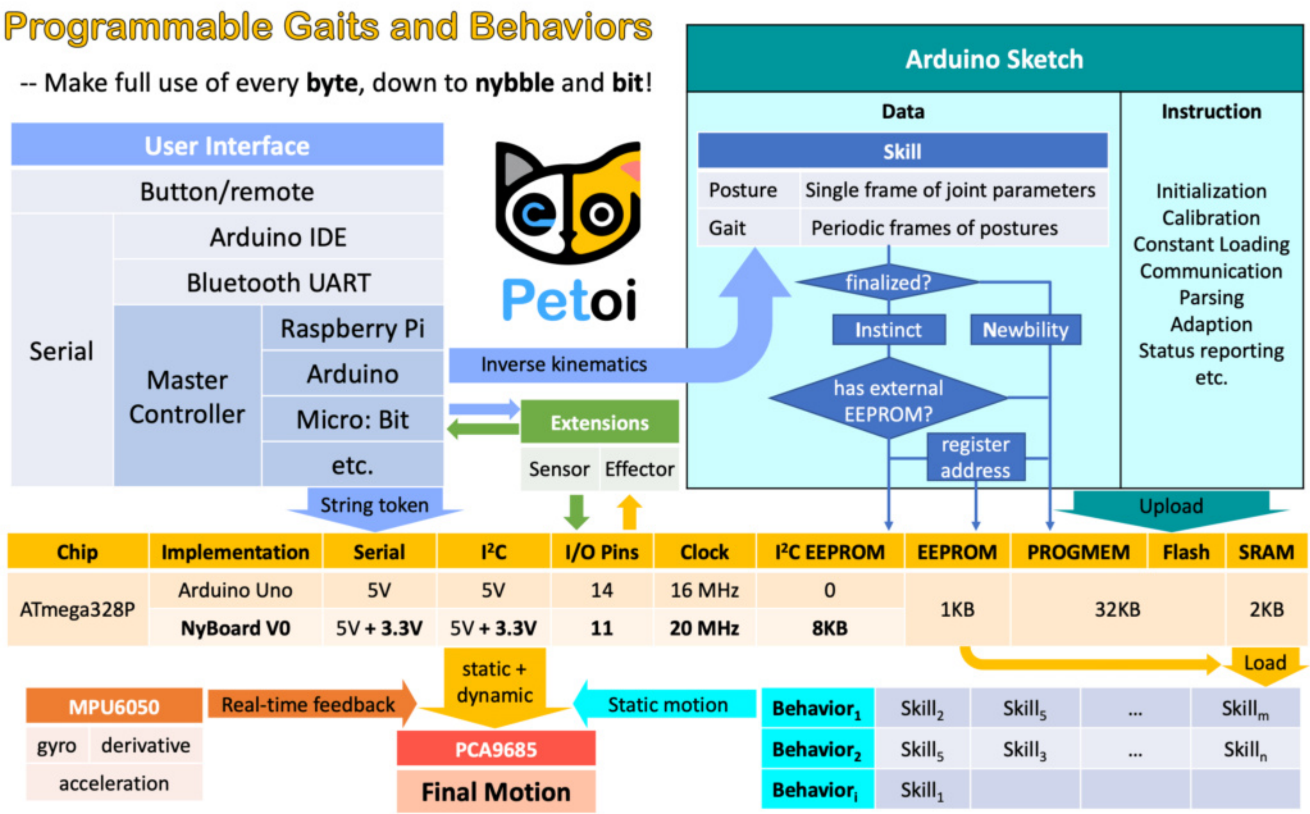
ound type	Occasion	Explaination
hort melody	Power up or reboot	The program starts successfully
hort beep	During use	The program receives a command
epetitive 3 beeps	During use and pausing the movements	The battery is low or unconnected

- If Nybble keeps beeping after you run **OpenCat.ino**, with numbers printed in the serial monitor, it's the low voltage alarm being triggered. You need to power NyBoard with two 3.7V Li-ion/Li-poly batteries to pass the threshold.
- "FIFO overflow! Using last reading!" in the serial monitor is an algorithmic fix to the original MPU6050 library. It's not a bug.
- The servos are designed to be driven by internal gears. Avoid rotating the servos too fast from outside.
- **Don't keep Nybble walking for too long.** That will overheat the electronics and reduce the servos' life span. If you have NyBoard V0_1 (our oldest version), It's possible to reconfigure it to make Nybble more stable.
- Sometimes the program may halt due to voltage fluctuation. Check if the battery is running low (< 3.5V each or <7.0V in series). Press the reset button on NyBoard to restart the program.
- Nybble has acrophobia! If you lift it and rotate it over a certain degree, its current movement will be interrupted. Don't flip Nybble over to scare it!
- Be kind as if you were playing with a real kitten. (^=◕ᴥ◕=^)

8 Teach Nybble New Skills

"Give a cat a fish and you feed it for a day. Teach a cat to fish and you feed it for a lifetime."

8.1. Understand skills in InstinctNybble.h.



EEPROM has limited (1,000,000) write cycles. So I want to minimize the write operations on it.

There are two kinds of skills: **Instincts** and **Newbility**. The addresses of both are written to the onboard EEPROM(1KB) as a lookup table, but the actual data is stored at different memory locations:

- I2C EEPROM (8KB) stores **Instincts**.

The Instincts are already fine-tuned/fixed skills. You can compare them to “muscle memory”. Multiple Instincts are linearly written to the I2C EEPROM only once with **WriteInstinct.ino**. Their addresses are generated and saved to the lookup table in onboard EEPROM during the runtime of **WriteInstinct.ino**.

- PROGMEM (sharing the 32KB flash with the sketch) stores **Newbility**.

A Newbility is any new experimental skill that requires a lot of tests. It's not written to the I2C nor onboard EEPROM, but the flash memory in the format of PROGMEM. It has to be uploaded as one part of the Arduino sketch. Its address is also assigned during the runtime of the code, though the value rarely changes if the total number of skills (including all Instincts and Newbilities) is unchanged.

8.2. Example InstinctNybble.h

```
//a short version of Instinct.h as example

#define WalkingDOF 8
#define NUM_SKILLS 4
#define I2C_EEPROM

const char rest[] PROGMEM = {
1, 0, 0, 1,
-30,-80,-45,  0, -3, -3,  3,  3, 60, 60,-60,-60,-45,-45, 45, 45,};
const char zero[] PROGMEM = {
1, 0, 0, 1,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,};

const char cr[] PROGMEM = {
26, 0, -5, 1,
 35, 37,-46,-53,-23,-32, -3, 12,
 40, 28,-42,-59,-24,-28, -4, 12,
 ...
 33, 39,-47,-51,-22,-32, -3, 11,
};

const char pu[] PROGMEM = {
-5, 0, -15, 1,
 1, 2, 3,
   30,  30,   0,   0,   0,   0,   0,   0,  60,  60,  70,  70,  15,  15, -70, -70,
    0, -40,   0,   0,   0,   0,   0,   0,  30,  30,  95,  95,  60,  60, -70, -70,
    5,   5,   0,   0,   0,   0,   0,   0,  75,  75,  55,  55, -50, -50, -75, -75,
    5,   5,   0,   0,   0,   0,   0,   0,  75, -70,  55,  55, -50,  70, -75, -75,
   60, -30, -45,   0,   0,   0,   0,   0,  70, -70,  55,   0, -30, -45, -75, -45,
};

#if !defined(MAIN_SKETCH) || !defined(I2C_EEPROM)
const char* skillNameWithType[] =
{"crI", "puI", "restI", "zeroN",};
const char* progmemPointer[] =
{cr, pu, rest, zero, };
#else
const char* progmemPointer[] = {zero};
#endif
```

8.2.1. Defined constants

`define WalkingDOF 8`

defines the number of DoF (Degree of Freedom) for walking is 8 on Nybble.

`define NUM_SKILLS 4`

defines the total number of skills is 4. It should be the same as the number of items in the list `const`

`char* skillNameWithType[]`

`define I2C_EEPROM`

Means there's an I2C EEPROM on NyBoard to save Instincts.

> ⓘ If you are building your own circuit board that doesn't have it, comment out this line. Then both kinds of skills will be saved to the flash as PROGMEM. Obviously, it will reduce the available flash space for functional codes. If there were too many skills, it may even exceed the size limit for uploading the sketch.

8.2.2. Data structure of skill array

One frame of joint angles defines a static **posture**, while a series of frames defines a sequential postures, such as a **gait** or a **behavior**. Observe the following two examples:

```
const char rest[] PROGMEM = { //posture
1, 0, 0, 1,
-30,-80,-45,  0, -3, -3,  3,  3, 60, 60,-60,-60,-45,-45, 45, 45,};

const char cr[] PROGMEM = { //gait
26, 0, -5, 1,
 35, 37,-46,-53,-23,-32, -3, 12,
 40, 28,-42,-59,-24,-28, -4, 12,
 ...
 33, 39,-47,-51,-22,-32, -3, 11,
};
```

They are formatted as:

	Total # of Frames	Expected Body Orientation		Angle Ratio	Indexed Joint Angles															
		Roll	Pitch		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1
rest	1	0	0	1	-30	-80	-45	0	-3	-3	3	3	60	60	-60	-60	-45	-45	45	4
cr	26	0	-5	1									35	37	-46	-53	-23	-32	-3	1
													...	...	...	...	...	...	...	..
													33	39	-47	-51	-22	-32	-3	1

rest is a static posture, it has only one frame of 16 joint angles. **cr** is the abbreviation for "crawl". It has 26 frames of 8 (or 12, depending on the number of walking DOF) joint angles that form a repetitive gait. Expected body orientation defines the body angle when the robot is conducting the skill. If the body is tilted from the expected angles, the balancing algorithm will calculate some adjustments. Angle ratio is used when you want to store angles larger than the range of -128 to 127. Change the ratio to 2 so that you can save those large angles by dividing 2.

ⓘ

A posture has only one frame, and a gait has more than one frames and will be looped over.

The following example is a behavior:

```
const char pu[] PROGMEM = { //behavior
-5, 0, -15, 1,
 1, 2, 3,
   30,  30,   0,   0,   0,   0,   0,   0,  60,  60,  70,  70,  15,  15, -70, -70,
    0, -40,   0,   0,   0,   0,   0,   0,  30,  30,  95,  95,  60,  60, -70, -70,
    5,   5,   0,   0,   0,   0,   0,   0,  75,  75,  55,  55, -50, -50, -75, -75,
    5,   5,   0,   0,   0,   0,   0,   0,  75, -70,  55,  55, -50,  70, -75, -75,
   60, -30, -45,   0,   0,   0,   0,   0,  70, -70,  55,   0, -30, -45, -75, -45,
};
```

pu is short for "push up", and is a "behavior". Its data structure contains more information than posture and gait.

The first four elements are defined the same as before, except that the number of frames is saved as a negative value to indicate that it's a behavior. The next three elements define the repeating frames in the sequence: starting frame, ending frame, looping cycles. So the **1, 2, 3** in the example means the behavior should loop from the second to the third frame three times (the index starts from zero). The whole behavior array will be executed only once, rather than looping over like the gait.

Each frame contains 16 joint angles, and the last 4 elements define the speed of the transition, and the delay condition after each transition:

1. The default speed factor is 4, it can be changed to an integer from 1 (slow) to 127 (fast). The unit is **degree per step**. If it's set to 0, the servo will rotate to the target angle by its maximal speed (about 0.07sec/60 degrees). It's not recommended to use a value larger than 10 unless you understand the risks.
2. The default delay is 0. It can be set from 0 to 127, the unit is **50 ms**.
3. The 3rd number is the trigger axis. If it's not 0, the previous delay time will be ignored. The trigger of the next frame will depend on the body angle on the corresponding axis. 1 for the pitch axis, and 2 for the roll axis. The sign of the number defines the direction of the threshold, i.e. if the current angle is smaller or larger than the trigger angle.
4. The 4th number is the trigger angle. It can be -128 to 127 degrees.

8.2.3. Suffix for indicating Instinct and Newbility

You must upload **WriteConst.ino** to have the skills written to EEPROM for the first time. The following information will be used:

```
const char* skillNameWithType[] =
{"crI", "puI", "restI", "zeroN",};
const char* progmemPointer[] =
{cr, pu, rest, zero, };
```

Notice the suffix **I** or **N** in the skill name strings. They tell the program where to store skill data and when to assign their addresses.

Later, if the uploaded sketch is the main sketch **OpenCat.ino**, and if you are using NyBoard that has an I2C EEPROM, the program will only need the pointer to the Newbility list

`const char* progmemPointer[] = {zero};`

to extract the full knowledge of pre-defined skills.

8.3. Define new skills and behaviors

8.3.1 Modify the existing skill template

There's already a skill called "zeroN" in InstinctNybble.h. It's a posture at the zero state waiting for your new definition.

You can first use the command `mIndex Offset` to move an individual joint to your target position, then replace the joint angles (bold fonts) in array at once:

```
const char zero[] PROGMEM = {
1, 0, 0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,};
```

Because it's declared as a Newbility and doesn't require writing to I2C EEPROM, you can simply upload **OpenCat.ino** every time you change the array (without uploading **WriteInstinct.ino**). You can trigger the new posture by pressing **7>** on the IR remote, or type `kzero` in the serial monitor.

You can rename this skill, but remember to update the keymap of the IR remote.

8.3.2. Add more skills in InstinctNybble.h

You can add more skills in InstinctNybble.h. Remember to increase the skill number at the beginning of the file and add the corresponding skill name and pointer in the skill list array.

A skill can be called from the serial monitor with the token 'k' command. For example, `ksit` will move Nybble to posture "sit".

You can also tune new skills by sending posture frames through the serial port, using the **m, i, l** tokens without uploading a new sketch. After fine-tuning the skill, you can save it in the instinctNybble.h and upload it to the board as a newbility or instinct.

This git repo is a good starting point if you want to develop a customized gait.

Always check the actual code for the available skill names. We may alter the skill set as we iterate the software.

8.3.3 Automation

So far Nybble is controlled by the infrared remote. You make decisions for Nybble's behavior.

You can connect Nybble with your computer or smartphone, and let them send out instructions automatically. Nybble will try its best to follow those instructions.

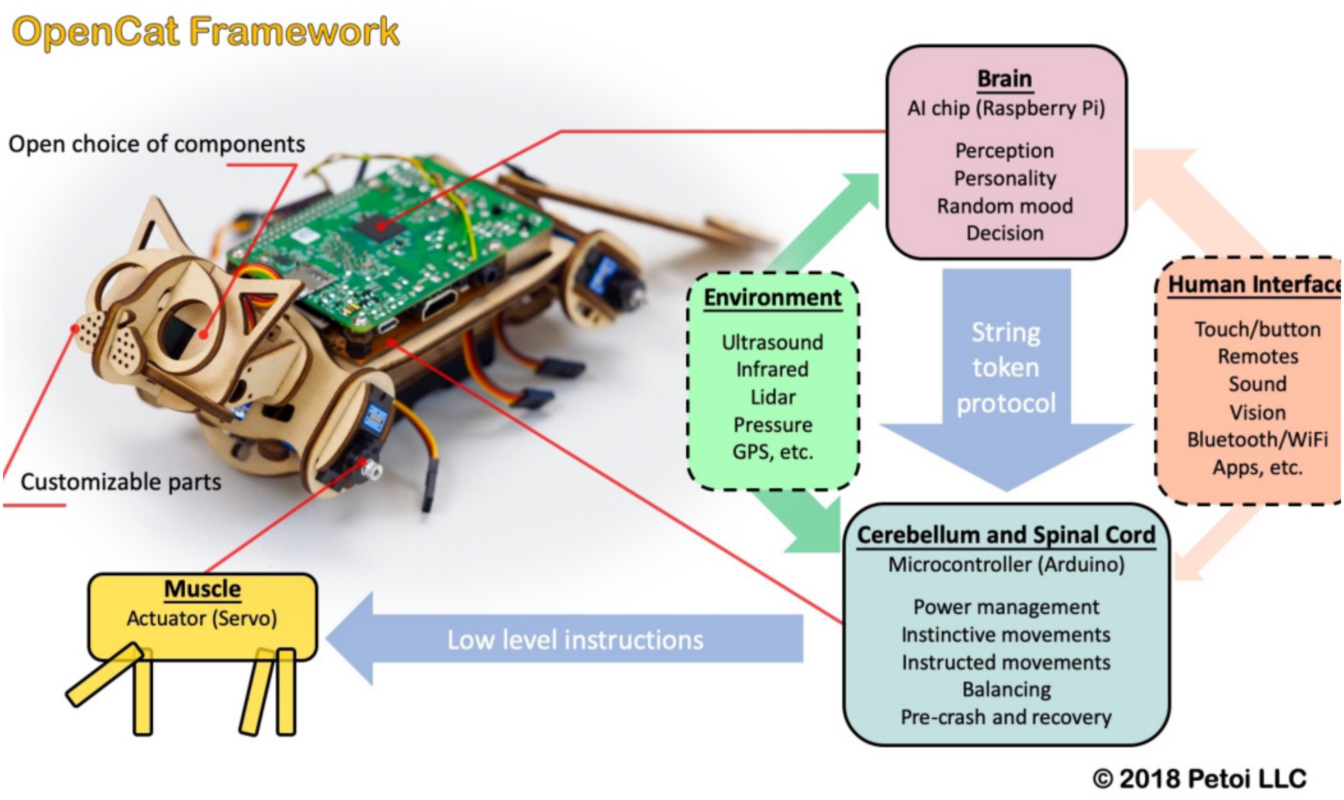
By adding some sensors (like a touch sensor), or some communication modules (like a voice control module), you can bring new perception and decision abilities to Nybble. You can accumulate those automatic behaviors and eventually make Nybble live by itself!

9 Understand OpenCat.h

"Let the cat out of the bag."

It will make another textbook.

The controlling framework behind Nybble is OpenCat, which I've been developing for a while. You can read more stories from my posts on Hackster.io.



It's too much work for now but you are welcome to discuss it with me on the forum or through email. Good thing is, I will keep the code compatible with future OpenCat models, and even your own DIY robots! Hopefully, the documentation will be completed during the processes.



10 ▯ Mess Up with the Robot

"Faster! Smarter! Cuter!" ▯

To be written by ▯ YOU!

Share your knowledge and creativity with the community at https://www.petoi.com/forum.





Useful Links